VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2020 Jan Svoboda

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

PŘESNÁ SEGMENTACE OBRAZOVÝCH DAT
PRECISE SEGMENTATION OF IMAGE DATA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Jan Svoboda**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Jan Mikulka, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Audio inženýrství**
specializace Zvuková produkce a nahrávání
Ústav telekomunikací

Student: Jan Svoboda ***ID:*** 200745
Ročník: 3 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Přesná segmentace obrazových dat

POKYNY PRO VYPRACOVÁNÍ:

V prostředí 3D Slicer zpracujte dostupná MR a CT data torza lidské páteře s okolním svalstvem i bez svalstva. Cílem zpracování je obrazová koregistrace, převzorkování a segmentace. V rámci bakalářské práce bude provedena implementace zásuvného modulu do prostředí 3D Slicer zajišťující potřebnou obrazovou analýzu a bude sestavena sada obrazů jednotlivých částí páteře obsahující koregistrovaná MR a CT obrazová data a odpovídající binární masky umožňující další zpracování jako učení pokročilých technik s prvky umělé inteligence.

DOPORUČENÁ LITERATURA:

[1] GONZALEZ, R. C.; WOODS, R. E.: Digital Image Processing, Prentice Hall, New Jersey, 2002.

[2] BRADSKI, G.; KAEHLER, A.: Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, Inc. USA 2008, ISBN: 978-0-596-51613-0.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 8.6.2020

Vedoucí práce: doc. Ing. Jan Mikulka, Ph.D.

doc. Ing. Jiří Schimmel, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práce pojednává o tvorbě rozšiřujícího modulu pro platformu 3D Slicer. Jádro modulu je implementací klasifikátoru Support Vector Machines, jenž je využíván k segmentaci obrazových dat. Obrazová data byla poskytnuta lékaři z Fakultní nemocnice Brno Bohunice. Jedná se obrazové sekvence páteře pořízené pomocí dvou modalit, tedy výpočetní tomografie a magnetické rezonance. Cílem práce bylo tato data zpracovat, konkrétněji pak provést převzorkování, prostorové srovnání a následnou registraci. Obrazy pořízené výpočetní tomografií pak díky svému dostatečnému kontrastu poskytují autorům možnost lépe odhadnout segmentaci kostní tkáně od okolního prostředí i pro obrazy pořízené pomocí magnetické rezonance dosahujících menšího kontrastu. Pro kontrastní obrazy pořízené pomocí výpočetní tomografie bylo dosaženo odpovídajících výsledků, u predikovaných masek méně kontrastních obrazů se tento nedostatek projevil hrubými nepřesnostmi.

KLÍČOVÁ SLOVA

3D Slicer, CT, metoda podpůrných vektorů, MRI, Python, registrace, segmentace, strojové učení, zpracování obrazu

ABSTRACT

The concern of this thesis is a development of an extension module for 3D Slicer platform. The core of the module is an implementation of a Support Vector Machines classifier, which is used for segmentation of the vertebral column image data provided by the University Hospital Brno. One of the goals of the thesis was resampling and registration of these image sequences. CT volumes provided solid contrast and were used as a reference for gaining properly segmented groups of vertebrae. Due to the low quality of the MRI volumes image data, segmentation of MRI images was not completely succesful. The extension module scripted in Python language can be seen as a tool and can be used in the future for different datasets.

KEYWORDS

3D Slicer, image processing, machine learning, Python, registration, segmentation, Support Vector Machines

SVOBODA, Jan. *Přesná segmentace obrazových dat*. Brno, 2020, 62 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Jan Mikulka, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Přesná segmentace obrazových dat“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Tímto bych rád poděkoval panu doc. Ing. Janu Mikulkovi, Ph.D. za odborné vedení práce, trpělivost a vždy pohotovou odezvu v rámci naší komunikace.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Tato práce se věnuje problematice zpracování obrazu, zejména pak segmentačním technikám aplikovaných v medicínské oblasti. Segmentací se rozumí vymezení požadované části obrazu, respektive rozdělení obrazové scény na jednotlivé objekty. Typicky se jedná o problém oddělení požadované anatomické struktury od okolního pozadí. Automatizace těchto technik oproti manuální segmentaci umožňuje nemalé šetření času při diagnostice. Velký důraz je ovšem kladen na přesnost, která je v lékařských aplikacích naprosto kritická. K získání vypovídajících výsledků segmentačních algoritmů je také nutné volit vhodné techniky předzpracování obrazových dat.

Základní předzpracování obrazových dat lze provést aplikací filtrace, která je ve své nejjednodušší formě schopná odstranit šum či zvýraznit hrany. Uniformitu dat, která je rozhodující zejména pro techniky strojového učení, lze dosáhnout shodným rozlišením všech obrazů. V případě nutné změny velikosti obrazu je použito převzorkování. Dalším termínem použitým v této práci je registrace obrazových dat, tedy prostorové srovnání obrazů, jež nachází využití při diagnozách nádorových onemocnění, radioterapii či obrazem vedených operací [1]. Problematika zmíněných termínů je dále popsána v druhé části kapitoly 1.

Obrazová data lidského torza, jejichž vlastnosti jsou prostudovány v experimentální části, byla pořízena pomocí MRI (Magnetic Resonance Imaging – zobrazování magnetickou rezonancí) a CT (Computed Tomography – výpočetní tomografie). Popisu činnosti těchto zobrazovacích zařízeních je věnována první část kapitoly 1. Pro následné zobrazení obrazů je používán program 3D Slicer, jenž se svými interními moduly představuje komplexní nástroj pro práci s medicínskými obrazy. V třetí části kapitoly 1 je společně s popisem interních knihoven rozebrána architektura tohoto softwaru. V neposlední řadě jsou zde také popsány jednotlivé způsoby implementace rozšiřujících zásuvných modulů.

V experimentální části je věnována pozornost již zmíněným obrazovým datům liského torza, konkrétně páteři a jejím jednotlivým obratlům. Kromě patřičných vlastností obrazových dat, jsou zde popsány i úskalí, kterými je soubor obrazových dat zatížen, a také je zde popsán průběh registrace obrazů. Následně je uveden postup samotné tvorby rozšiřujícího modulu pro program 3D Slicer, který realizuje segmentaci obrazových dat pomocí klasifikátoru SVM (metoda podpůrných vektorů – Support Vector Machines). Modul a příslušné skripty jsou implementovány v jazyce Python.

Samotnou motivací této práce je spolupráce s Fakultní nemocnicí v Brně Bohunicích, jež poskytla poměrně unikátní obrazová data. Tato jedinečnost spočívá v charakteru obrazových dat – byly pořízeny sekvence torza s měkkými tkáněmi a zároveň torza bez měkkých tkání, a to pomocí dvou modalit (CT a MRI), jak již bylo naznačeno. Při vhodné registraci kontrastních CT snímků páteře bez tkáně se snímky pořízenými pomocí MRI, která tradičně lépe zobrazuje měkké tkáně než kosti, a po následné segmentaci CT snímků můžeme získat binární masku samotné páteře pro obě modality. Budeme tedy znát také výsledek segmentace u obrazů pořízených MRI, což je velmi žádoucí, neboť provést optimální segmentaci kostní struktury u nekontrastních MRI obrazů je velmi složitý úkol. Nicméně předpokladem pro dosáhnutí tohoto výsledku je velmi přesná registrace obrazových sekvencí mezi oběma modalitami. V průběhu textu se bude k tomuto faktu přihlížet několikrát, jelikož konečné řešení spjaté s prostorovou orientací obrazů je u části vybraných obrazových sekvencí spíše orientační, a to z důvodu samotné kvality a vlastnosti obrazových dat. Avšak při dosáhnutí registrace a segmentace odpovídající kvality, mohou být binární masky CT obrazů použity jako měřítko úspěšnosti vyvíjených algoritmů pro segmentaci MRI obrazů.

1 Teoretická část

1.1 Zobrazovací přístroje

Zobrazovací přístroje v lékařství jsou velmi užitečnou nápomocnou silou při každodenní práci lékařů, pomáhají zobrazovat vnitřní struktury částí lidského těla pro následnou diagnostiku. Velkým krokem pro zdravotnictví byl nástup výpočetní techniky a digitalizace. V sedmdesátých letech minulého století, kdy bylo představeno první CT, jež i přes nízké rozlišení výstupních obrazových dat, představovalo slibný budoucí vývoj, začaly digitální zobrazovací zařízení transformovat moderní lékařství. S rostoucím výpočetním výkonem roste diagnostická užitečnost těchto zařízení a snižuje se čas celkového výpočtu, který je v oblastech jako traumatologie významným faktorem. Samotné zobrazení můžeme definovat jako převod určité fyzikální veličiny na jasovou hodnotu, jedná se tedy o převedení reálné scény do scény obrazové. Jednotlivé metody využívají různé parametry, které poté určují přednosti daného zobrazovacího systému.

1.1.1 Výpočetní tomografie

Jedná se o zobrazovací metodu, která pomocí rentgenového záření zobrazuje tělo pacienta v podobě souboru řezů. Díky pořízení snímků z různých směrů je možné rekonstruovat prostorovou obrazovou scénu. Ve své nejjednodušší formě je tato zobrazovací metoda založena na zdroji rentgenového záření (rentgenka – X–ray tube) a scintilačních detektorech na straně opačné. Celý tento systém v čase obíhá kolem pacienta, který je umístěn na posuvném lůžku a jehož tělo absorbuje záření. Tkáně pohlcují rentgenové záření rozdílně. Princip rentgenky spočívá ve vakuu umístěné katodě emitující elektrony směrem k anodě, které jsou při dopadu na povrch anody brzděny a vytvářejí mimo jiná záření i charakteristické rentgenové záření.

CT tedy měří prostorovou distribuci útlumového koeficientu $\mu(x, y)$ v určitých obrazových souřadnicích. V praxi se používají Hounsfieldovy jednotky (HU, CT čísla), jež jsou definovány vyjádřením denzity, tedy míry absorpce a rozptylu záření, pro jednotlivé prostorové pixely (voxely),

$$D = \frac{\mu_{\mathrm{T}} - \mu_{\mathrm{w}}}{\mu_{\mathrm{w}}} \cdot \mathrm{k}, \tag{1.1}$$

kde μ_{T} je koeficient zeslabení tkáně, μ_{w} koeficient zeslabení vody a k je smluvená konstanta o velikosti 1000. Hodnoty se typicky pohybují v rozmezích -1000 HU až 1000 HU. Nejnižších hodnot nabývá vzduch, nejvyšších hodnot pak kompaktní kosti. CT disponuje vysokým kontrastem obrazů, zjevnou nevýhodou je pak vystavení pacienta určitému škodlivému záření. Problematice výpočetní tomografie,

její historii, jednotlivým konstrukčním řešením a způsobům vytváření výsledného obrazu se věnuje literatura [2].

1.1.2 Magnetická rezonance

Další používanou metodou pro neinvazivní vyšetření je MRI. Fyzikálním principem je působení velmi silného magnetického pole, které ovlivní mechanickou vlastnost, konkrétně spin vodíkových jader. Pro spiny je z hlediska energie výhodné se vůči tomuto magnetickému poli orientovat paralelně. Pro dosažení vhodné orientace je spin rotován. Za těchto podmínek je možné dodávat spinu energii v podobě elekromagnetického impulsu, energie je absorbována a dochází k excitovanému stavu. Při odeznění pulzu dochází k relaxaci – k návratu do rovnovážného stavu. Tyto zmíněné procesy lze detekovat vhodnou přijímací cívkou. Při volbě určité periody opakování excitačních pulzů dostáváme a dále vhodným nastavením času mezi excitací a měřením přijímaného signálu dostáváme T_1–vážený obraz či T_2–vážený obraz. Charakter látky tkáně poté rozhoduje o konstantách T_1 a T_2 a umožňuje rozlišení jednotlivých typů tkáně. MRI je významnou zobrazovací metodou měkkých tkání, což je předněji dáno obsahem vody těchto částí těla. Výhodou je také šetrnost metody vůči pacientovi, který není vystaven škodlivému záření. Nevýhodou je přítomnost silného magnetického pole, které neumožňuje vyšetření pacientů s kovovými náhradami a implantáty v těle (např. kardiostimulátory), poté delší čas vyšetření a v neposlední řadě vysoká pořizovací cena.

1.2 Zpracování obrazových dat

Na obraz se můžeme dívat jako na dvourozměrný signál. Typicky je vyjádřen obrazovou funkcí, které je závislá na jednotlivých souřadnicích x, y:

$$f(x, y), \tag{1.2}$$

hodnotou této funkce je poté vyjádření jasu – veličiny, která odpovídá vnímání obrazu člověkem. Pro digitální, též nespojité zpracování obrazu je nutné analogový (spojitý) obrazový signál nutné vzorkovat a převést ho do oblasti digitální. Ve skutečnosti se jedná o ztrátovou metodu, kdy dochází k potlačení informace, která je navíc zatížena kvantizační chybou. Z analogového obrazu tedy dostáváme konečný počet obrazových elementů – pixelů, které jsou nejčastěji uspořádány do matice o rozměrech M×N. Na jednotlivých pozicích se poté vyskytuje číslo, jež jednoduše reprezentuje jasovou hodnotu daného pixelu. Zpracování obrazu ve své základní formě vyžaduje znalost matematiky, zejména pak lineární algebry a matematické analýzy, pro pokročilejší aplikace na vyšších úrovních počítačového vidění je poté

nutná znalost metod strojového učení. Literatura [3] rozděluje problematiku obrazu na následující části:

- snímání, digitalizace a uložení obrazu v počítači,
- předzpracování,
- segmentace obrazu na objekty,
- popis objektů,
- porozumění obsahu obrazu (zejména klasifikace objektů).

1.2.1 DICOM

Ve zdravotnictví existuje několik standardů pro archivaci, sdílení či samotný tisk obrazových dat. Tato standardizace umožňuje efektivní komunikaci napříč zdravotnickými zařízeními. Jedním ze standardů pro práci s obrazovými daty je DICOM (Digital Imaging and Communications in Medicine), který je definován příponou .dcm. Obsah tohoto formátu je kromě obrazových dat vybavený i informacemi o pacientovi, době a místě pořízení či dodatečný popis vlastností obrazu. Objemová data jsou uložena jako skupina jednotlivých dvourozměrných řezů. Pro více informací a zjištění novinek je možné navštívit webovou stránku[1].

1.2.2 Registrace

Obrazová registrace spočívá ve srovnání obrazů v prostoru vůči obrazu referenčnímu. Pro vhodné vykonání registrace je jako referenční obraz zvolen takový obraz, jenž poskytuje nejkvalitnější vlastnosti – často se jedná o obraz s nízkým šumem či s vyšším rozlišením. Cílem je tedy najít vhodnou transformaci vstupního obrazu, tak aby jeho výsledná podoba odpovídala referenčnímu obrazu. K popisu obrazů můžeme použít vztahy [4]:

$$\mathbf{X}_{\mathrm{A}} = \{\mathbf{x}_{\mathrm{A}}\}, \mathbf{X}_{\mathrm{B}} = \{\mathbf{x}_{\mathrm{B}}\}, \mathbf{X}_{\mathrm{A}} \mapsto \{\mathbf{A}(\mathbf{x}_{\mathrm{A}})\}, \mathbf{X}_{\mathrm{B}} \mapsto \{\mathbf{B}(\mathbf{x}_{\mathrm{B}})\}, \tag{1.3}$$

kde $\mathbf{x}_{\mathrm{A}}$ je vektor popisující zorné pole $\mathbf{X}_{\mathrm{A}}$ obrazu $\mathbf{A}$, který je transformován, $\mathbf{x}_{\mathrm{B}}$ je vektor popisující zorné pole $\mathbf{X}_{\mathrm{B}}$ obrazu $\mathbf{B}$, jenž je referencí. Dochází k mapování k odpovídajícím hodnotám intenzity $\mathbf{A}(\mathbf{x}_{\mathrm{A}}), \mathbf{B}(\mathbf{x}_{\mathrm{B}})$. Uvedený problém geometrické transformace můžeme zapsat pomocí vztahů:

$$T_{\alpha} : \mathbf{x}_{\mathrm{A}'} = T_{\alpha}(\mathbf{x}_{\mathrm{A}}), \mathbf{A}'(\mathbf{x}_{\mathrm{A}'}) = \mathbf{A}(\mathbf{x}_{\mathrm{A}}), \mathbf{x}_{\mathrm{A}'} = \mathbf{x}_{\mathrm{B}}, \tag{1.4}$$

kde $\mathbf{A}'$ značí již transformovaný obraz, T_{α} značí geometrickou transformaci, α pak vektor této transformace. V ideální případě po provedení transformace je transformovaný obraz $\mathbf{A}'$ srovnán (překryt) s referenčním obrazem $\mathbf{B}$. Tento výsledek však

[1] https://www.dicomstandard.org

není vždy možný – překrývá se jen určitá část obrazů. Pomocí optimalizace poté hledáme takové parametry transformace, která splní námi požadované vlastnosti, respektive se jedná o proces nalezení maxima či minima kriteriální funkce, záleží zde na charakteru porovnávacích parametrů [4].

Samotnou registraci je tedy možné provést podle několika určujících kritérií. Naším úkolem je tedy najít takové aspekty obrazových dat, které budou vypovídající o podobnosti souboru obrazů, a umožní tak optimální řešení registrace. Pro registraci v dvourozměrném prostoru existují metody, jež jsou založeny na několika významných bodech, které jsou vybrány z dostupných obrazů, dále pak metody pracující na základě srovnání segmentovaných binárních masek či metody přímo pracující s jasovou hodnotou daného voxelu [5].

Geometrická transformace

V následující části textu je také vhodné popsat typy geometrické transformace, jež se nemalou částí podílejí na výsledné registraci souboru obrazů. Princip transformace se skrývá v přepočtu polohy každého body. V diskrétní oblasti pak vyvstává problém výsledné polohy daného bodu, který může ležet mimo obrazový rastr. Je tedy nutné aproximovat výslednou jasovou hodnotu na celočíselné pozici. Touto problematikou se zabývá kapitola o převzorkování a interpolaci. Přehledný popis typů geometrické transformace pak poskytuje článek [6].

Rigidní transformace zachovává vzdálenost mezi dvěma body [7]. Úkolem je najít vhodnou Euklidovskou transformaci obrazu $\mathbf{A}$ ve tvaru [8]:

$$T^*(\mathbf{x}_\mathrm{A}) = \mathbf{R}\mathbf{x}_\mathrm{A} + \mathbf{t}, \tag{1.5}$$

kde $T^*(\mathbf{x}_\mathrm{A})$ symbolizuje transformaci obrazu $\mathbf{A}$ při hledání maximální podobnosti vůči referenčnímu obrazu $\mathbf{B}$, $\mathbf{R}$ je transformační matice a $\mathbf{t}$ vektor určující samotný posun. Jedná se tedy o lineární vztah. Rigidní transformace zahrnuje rotaci, translaci, zrcadlení či jejich kombinaci. Důležité je nahlížet na obraz jako na neměnný objekt, který můžeme posouvat či rotovat v prostoru, nemůžeme však měnit jeho rozměry či tvar. Využití rigidní transformace spočívá ve srovnání obrazů, které se od sebe liší jen nepatrně [6]. To je typické pro sérií obrazů MRI, jež byly pořízeny v časové návaznosti.

Afinní transformace je charakteristická tím, že zachovává paralelnost přímek, nikoliv však jejich délky či úhly. Rozšiřuje tedy možnosti rigidní transformace o změnu velikosti či zkosení.

Projektivní transformace nezachovává výše zmíněnou paralelnost přímek, přesto mapuje přímky na přímky. Snaží se zachytit, jak se pozorovaný předmět jeví při změně pozice pozorovatele. Může tedy vytvářet zkreslení perspektivy.

Jednotlivé operace je možné vyjádřit pomocí matic, jež operují s homogenními souřadnicemi v rovině $\mathbb{R}^2$. Uvažujme tedy bod $X = (x_1, x_2)$, kde x_1 a x_2 popisují umístění tohoto bodu v rovině. Pro vyjádření bodu v homogenních souřadnicích se nejčastěji používá zápis $[x_1, x_2, 1]^{\mathrm{T}}$ [3].

Posunutí poté můžeme realizovat pomocí následující matice [9]:

$$\begin{pmatrix} x_1' \\ x_2' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix},$$

kde x_1' a x_2' představují nové souřadnice bodu X, který je posunut o odpovídající koeficienty a a b. Pro rotaci bodu X využijeme maticový zápis uvedený níže, kde bod X rotujeme kolem počátku $S = (0, 0)$ o úhel α:

$$\begin{pmatrix} x_1' \\ x_2' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}.$$

Změnu měřítka můžeme uskutečnit jednoduchým vynásobením příslušné souřadnice konstantou. Této skutečnosti odpovídá maticový zápis:

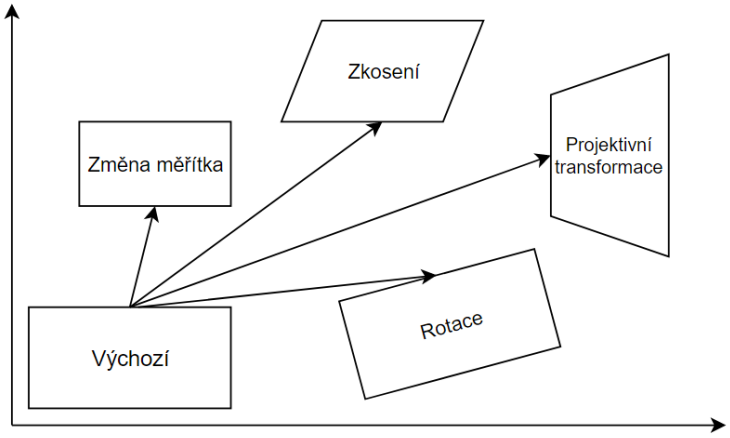
$$\begin{pmatrix} x_1' \\ x_2' \\ 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}.$$

Operace zkosení přiřazuje bodu X souřadnice x_1' a x_2' pomocí následujících vztahů $x_1' = x_1 + ax_2$, $x_2' = bx_1 + x_2$, kde a, b jsou koeficienty určující samotné zkosení. Maticově lze zkosení poté vyjádřit takto:

$$\begin{pmatrix} x_1' \\ x_2' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & a & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}.$$

Velmi důležitou skupinou transformací, které ovšem přesahují rámec této práce, jsou transformace nelineární. Jejich základní myšlenkou je hledat vhodný způsob změny tvaru samotného objektu. Pomocí deformace tak často dospějí k mnohem více vypovídajícím výsledkům než transformace lineární. U některých typů transformací se však pracuje se složitější matematickými konstrukcemi, jejichž výzkumu

je věnována značná pozornost. Také je nutné zmínit vyšší časovou náročnost, která může být u klinických aplikací značnou překážkou. Jednou z transformací, která je i přes svou nelineárnost poměrně dobře časově optimalizovaná, je transformace typu spline. Tato transformace se snaží o co nejjemnější reprezentaci cílové křivky pomocí aproximační funkce. Její výsledky jsou prezentovány v článku [10], kde oproti afinní transformaci dosahuje daleko lepší kompenzace deformace objektu v pohybu. Elastická transformace v článku [11] dosahuje taktéž uspokojivých časových výsledků. Autoři zavádí možnost přidání významných bodů do obrazu, které v případě složitých problémů dopomáhají úspěšnosti metody.



Obr. 1.1: Přehled jednotlivých transformací

1.2.3 Převzorkování

V oblasti zpracování obrazu se převzorkováním myslí změna rozměrů obrazu či jeho rozlišení. Je důležité mít na paměti, že převzorkováním obrazu na nižší rozlišení se ztrácí informace. Nicméně při některých metodách – typicky u strojového učení – je použití převzorkování často nezbytné. Pro účely zrychlení výpočtu se tedy obraz převzorkuje na nižší rozlišení, a to takovým způsobem, aby došlo co k nejmenšímu potlačení důležité informace. Jelikož při převzorkování manipulujeme s pixely obrazu, tak poté podobně jako u geometrické transformace narážíme na problém výpočtu intenzity v celočíselných souřadnicích. Ta se aproximuje pomocí interpolační funkce, která na základě jasových intenzit okolních bodů dopočte intenzitu v bodě, jehož intenzita díky nespojitosti diskrétní jasové funkce nebyla definována. Přehled interpolačních funkcí je uveden v následující části textu.

Interpolace

Problém interpolace spočívá v nalezení funkce $q(x), x \in [a, b]$, takové, že je splněno $q(x_i) = f(x_i), i = 1, \ldots, n$, kdy uvažujeme uzavřený interval $[a, b], a \in \mathbb{R}, b \in \mathbb{R}$ a body $x_1, \ldots, x_n, n \in \mathbb{N}$, takové, že platí $a \leq x_1 < \cdots < x_n \leq b$. Hodnoty v jednotlivých bodech jsou pak dány předpisem $f(x_1), \ldots, f(x_n)$ [9]. Článek [13] popisuje nejvyužívanější metody interpolace obrazů a zároveň dodává, že samotná interpolace je nejčastěji realizována konvolucí obrazu s interpolační maticí či vektorem. Z celé řady různých interpolačních technik je vhodné zmínit interpolaci pomocí nejbližšího souseda, interpolaci bikubickou a interpolací bilineární. Pro jednoduchost budeme uvažovat práci s šedotónovým obrazem, poté je jas jasně určen jednou hodnotou, na rozdíl od barevného obrazu, kdy je jas vyjádřen vícerozměrnou diskrétní funkcí.

Metoda nejbližšího souseda je nejjednodušší interpolační metodou, jež jasové hodnotě transformovaného obrazu na pozici (x', y') přiřadí nám známou jasovou hodnotu nejbližšího bodu na pozici (x, y). Interpolaci nejbližším sousedem je také možné řešit vhodným zaokrouhlením souřadnic. Metoda disponuje jednoduchou implementací, nicméně její nevýhodou je zkreslení hran u výsledného obrazu, které je ve většině aplikací nežádoucí.

Sofistikovanější interpolační metodou je bikubická metoda, jež dosahuje jemnější aproximace jasové hodnoty pomocí kubických polynomů při uvažování většího počtu okolních bodů. Jemnost je vyžadována zejména v pokročilé grafice či právě u medicínských obrazů, jejichž interpolací se s potřebným matematickým základem zabývá článek [14].

Bilineární interpolační metoda pomocí lineárního vztahu dopočítá jas bodu o souřadnicích (x', y'), a to v závislosti na jasových hodnotách jeho čtyř sousedů. Vztah, který popisuje jasovou hodnotu šedotónového obrazu $v(x', y')$, lze zapsat v podobě:

$$v(x', y') = ax' + by' + cx'y' + d, \tag{1.6}$$

kde hodnoty čtyř koeficientů a, b, c, d získáme pomocí řešení čtyř rovnic o čtyřech neznámých [15]. Platí zde úměrnost vlivu čtyř sousedních bodů, a to taková, že čím je sousední bod blíže interpolovanému bodu, tím více se projeví jeho jasová hodnota do hodnoty výsledné. K lepší názornosti je dobré se odkázat na literaturu [12], kde je výstižně graficky zobrazena problematika interpolovaného obrazu.

1.2.4 Segmentace

Velmi důležitým termínem v oblasti zpracování obrazu je segmentace. Může být popsána jako operace rozdělení obrazu na jednotlivé objekty. Tento proces umožní z obrazu vyčlenit takové úseky, které jsou důležité pro další kroky zpracování. Ve své podstatě ji můžeme interpretovat jako nezbytný nástroj pro následné pokročilé techniky jako detekci či klasifikaci objektů v obraze. Výsledkem segmentace je obraz o původních rozměrech s vyznačenými oblastmi našeho zájmu – v medicínské oblasti se nejčastěji jedná o anatomické struktury oddělené od zbylých části obrazu. Současně se jednotlivý podobraz nepřekrývá s jinými segmentovanými podobrazy. Segmentované části jsou ve výsledném obrazu typicky vyznačeny pomocí oblasti s homogenní barvou. Ve své podstatě je takto vytvořena binární maska sdružující všechny obrazové body patřící k segmentovanému objektu. Matematicky můžeme problém segmentaci vyjádřit takto [16]:

$$\bigcup_{i=1}^{n} \mathbf{R}_{\mathrm{i}} = f(x, y), \tag{1.7}$$

kde $\mathbf{R_i}$ reprezentuje jednotlivé podobrazy segmentovaného obrazu popsaného funkcí $f(x, y)$. Zároveň platí vztah:

$$\mathbf{R}_{\mathrm{a}} \cap \mathbf{R}_{\mathrm{b}} = \emptyset, a \neq b, \tag{1.8}$$

který vyjadřuje již zmíněné nepřekrytí obrazů $\mathbf{R}_{\mathrm{a}}$ a $\mathbf{R}_{\mathrm{b}}$. Automatickým segmentačním metodám je v současnosti věnována velká pozornost, jelikož při vhodné optimalizaci vykazují chvályhodné výsledky a šetří důležitý čas [17]. Pro nenáročné aplikace lze použít primitivní metody založené na jednoduchých úvahách, pro aplikace zakládající si na přesnosti jsou mimo jiné využívány pokročilé techniky strojového učení. V následující části textu budou zmíněny vybrané segmentační metody. Širší přehled nabízí literatura [18].

Segmentace prahováním

Tato velmi jednoduchá metoda je založena na rozhodovací úrovni vztažené k jasu obrazu. Je stanovena určitá číselná hodnota a zároveň prohlášeno, že hodnoty jasu ležící pod touto hodnotou budou započítávány do pozadí a hodnoty jasu nacházející se nad hranicí budou zařazeny do popředí. Z přirozenosti vyplývá, že se jedná o velmi hrubou metodu, jejíž kritická část leží ve volbě vhodné číselné hranice. Optimální volbou se zabývá algoritmus Otsu [20], který pomocí histogramu (graf vyjadřující rozložení jasu v obrazu) a metod statistiky, zejména pak rozptylu σ^2, vypočítá vhodnou číselnou hranici.

Hranové detektory

Metody založené na detekci hran využívají změnu, respektive derivaci jasové funkce. Jádro metod spočívá v nalezení takových částí obrazu, kde se skokově mění již zmíněný jas obrazu. Při tomto předpokladu můžeme tvrdit, že jsme v rámci obrazu našli hranu či linii, která například odděluje segmentovaný objekt od pozadí. Pro detekci pomocí první derivace je opět stanoven číselný práh, který určuje zda změna vyjádřena gradientem je dostatečně významná. V praktické rovině svou roli hrají matice – konvoluční jádra filtrů. Tyto jádra, jejichž rozměr je typicky definován lichým číslem, jsou schopny provést detekci hran vhodným nastavením vah. Konvoluční jádra však dokáží realizovat detekci jen v jednom směru, proto se často uvádí pár konvolučních jader, kdy první jádro dokáže detekovat hrany vertikální a druhé jádro hrany horizontální. Nejpoužívanější jádra lze prohlédnout v publikaci [16].
Operace využívající druhou derivaci jasové funkce detekují průchod nulou. Výhodou je invariantnost vůči směru, jako zástupce můžeme zmínit operátor Laplacian, jehož impulzní odezvu můžeme reprezentovat jádrem $\mathbf{L}$:

$$\mathbf{L} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}.$$

Houghova transformace je speciálním případem hranových detektorů popisující geometrické vlastnosti hledaného objektu. Nejtypičtěji se jedná o detekování přímek v obraze. Přímku lze analyticky popsat pomocí vztahu:

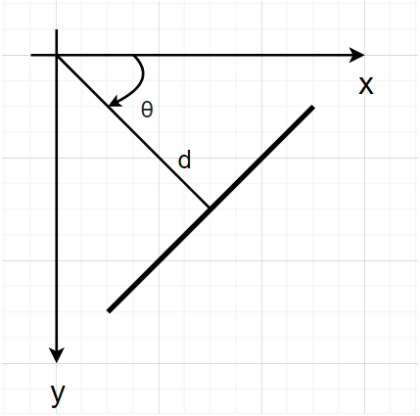
$$y = mx + b, \tag{1.9}$$

kde y a x jsou souřadnicemi bodů na přímce, m a b koeficienty určující sklon, respektive posunutí přímky. Tato rovnice však nevyhovuje přímkám ve svislém směru [19], je tedy nutné zavést vztah:

$$d = x\cos\theta - y\sin\theta, \tag{1.10}$$

kde d je vzdálenost přímky od počátku souřadnic a θ je úhel svíraný normálou přímky a horizontální osou. Ve své podstatě jsme převedli přímku do Houghova prostoru – přímka je popsána dvojicí parametrů d a θ. V případě hledání přímky v obrazu jsou vstupními daty souřadnice pixelů x a y, neznámými parametry pak d a θ. Po dosazení x, y do rovnice 1.10 získáváme množinu řešení, které se v Houghové prostoru znázorní jako křivka. Při dosazení dalších pixelů, respektive jejich souřadnic, které v obraze společně s dalšími pixely tvoří přímku, dochází v Houghově

prostoru k protnutí křivek v určitém bodě definovaným parametry d, θ. Bod, ve kterém se protínají křivky, poté popisuje hledanou přímku pomocí dvojice zmíněných parametrů d, θ.



Obr. 1.2: Přímka a její parametry využívané u Houghovy transformace

Metoda rozvodí

Metoda známá pod anglickým termínem watershed (rozvodí či povodí), poprvé publikována v sedmdesátých letech minulého století [21], se dívá na problematiku segmentace z hlediska geografické podobnosti. Obraz představuje jakýsi terén, kde maximální hodnoty jasové funkce tvoří pohoří a minimální hodnoty údolí. Tento reliéf je postupně zaplavován vodou pramenící v minimálních hodnotách jasové funkce, respektive v údolích. Jednotlivé prameny tak vytváří segmentované oblasti. V místech přechodu mezi jednotlivými regiony vzniká hranice – v naší analogii hráz – reprezentována křivkou. Ukončení algoritmu nastává, jakmile hladina dosáhne maximální výškové souřadnice (t.j. maximální jasové hodnoty). Výstupem této metody je velké množství segmentovaných oblastí, které již nemusí odpovídat reálnému objektu. Pro dosažení odpovídajících výsledků je nutné tyto oblasti postupně spojovat [22]. Výhodou metody je pak určitá odolnost vůči šumu.

Aktivní kontury

Metoda využívající aktivní kontury pracuje s počáteční křivkou, která je inicializována na začátku běhu algoritmu, a jejíž počáteční tvar může určit sám uživatel na základě znalosti obrazu, respektive oblasti obrazu, kterou chce segmentovat. Princip

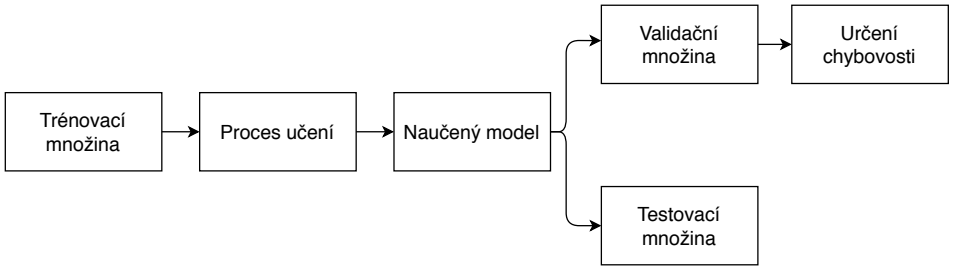
metody spočívá v deformaci této křivky takovým způsobem, aby v konečném stádiu tvořila obrys segmentovaného objektu. Deformovaná křivka je taktéž nazývaná anglickým termínem snake. Ve své podstatě je snake definován souborem vektorů opisující segmentovaný objekt. Na samotné deformaci se podílejí celkově tři složky energetické funkce E_{T} podle následujícího vztahu [23]:

$$E_{\mathrm{C}} = E_{\mathrm{i}} + E_{\mathrm{e}} + E_{\mathrm{c}}, \tag{1.11}$$

kde E_{i} se vztahuje k vnitřní energii zachovávající hladkost křivky, E_{e} je energie deformující křivku působící proti E_{i}, E_{c} je možné nadefinovat jako dodatečný požadavek deformace. Úkolem je tento vztah minimalizovat a dosáhnout tak optimálního obrysu objektu. Rozsáhlejší popis této problematiky je realizován v literatuře [17].

Strojové učení

V této části práce se dostáváme k popisu metod v oblasti strojového učení. Algoritmy této oblasti jsou tedy napsány tak, aby byl výsledný model schopen se učit. Samotné učení můžeme definovat jako přizpůsobení se vnějším podmínkách za účelem dosáhnutí co nejlepších výsledků v námi požadované oblasti. Není snad nutno zmiňovat, že daný proces učení vyžaduje oproti standardním algoritmům jistou časovou náročnost. Aby modely strojového učení bylo vůbec možné využít v praxi, nejde jen o investování času, ale je také nutné zajistit odpovídající kvalitu trénovacích dat. Tento postup je definován jako tzv. učení s učitelem – správný výstup v rámci testovacích dat je předem určený, program je poté pomocí definovaných mechanismů schopný upravit své vnitřní parametry tak, aby výstup programu odpovídal výstupu správnému. Literatura [24] definuje strojové učení jako snahu o optimální reprezentaci dat v předem definovaném prostoru. Pokud dokážeme optimálně reprezentovat vstupní data, je dosažení správného výstupu z pravidla snadnější. Celý tento postup můžeme znázornit podle následujícího diagramu.



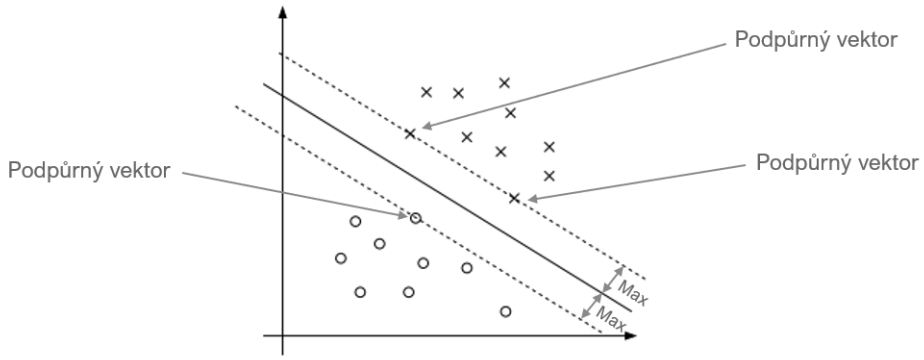
Obr. 1.3: Diagram průběhu strojového učení

Metody strojového učení staví zejména na matematických základech, důležitou roli také hraje časová a výpočetní optimalizace algoritmů. Pro samotné vyhodnocení

chybovosti naučeného modelu je využíváno metod statistiky. Úvodem do statistiky se zabývá literatura [25], jež poskytuje nezbytný základ statistických metod.

Support Vector Machines

Metoda podpůrných vektorů, označována jako SVM (Support Vector Machines), je metodou provádějící klasifikaci vstupních dat, tzn. rozdělení vstupů do jednotlivých tříd. Metoda SVM je definována jako binární klasifikátor – rozděluje vstupní data do dvou tříd [26]. Ve své podstatě se SVM snaží rozdělit prostor pomocí nadroviny tak, aby vektory jednotlivých tříd byly nadrovině co nejvzdálenější. Nadrovina je definována v prostoru o n dimenzích jako podprostor o $n-1$ dimenzích. V případě roviny je tedy nadrovinou přímka. Dalším důležitým termínem je podpůrný vektor reprezentovaný bodem v prostoru, jenž je ze všech vektorů dané třídy právě nejbližším vektorem k rozdělující nadrovině. Danou problematiku můžeme také interpretovat jako nalezení nejširšího hraničního pásma.



Obr. 1.4: Rozdělení roviny pomocí metody SVM

Problém klasifikace do jednotlivých tříd můžeme popsat pomocí funkce [27]:

$$f : \mathbb{R}^{\mathrm{N}} \rightarrow (\pm 1), \tag{1.12}$$

která přiřazuje $N-$rozměrná data $\mathbf{x}_{\mathrm{i}}$ do dané třídy y_{i}. Zároveň správně provádí klasifikaci pro novou dvojici $(\mathbf{x}, y)$, tedy $f(\mathbf{x}) = y$. Konkrétně pro metodu SVM při uvažovaní prostoru $\mathbb{R}^2$ můžeme rozdělení vstupních vektorů provést pomocí přímky definované:

$$\vec{w} \cdot \vec{x} + b = 0, \tag{1.13}$$

kde $\vec{w}$ reprezentuje normálu nadroviny, b poté koeficient posunutí. Kolmou vzdálenost nadroviny od počátku souřadnic vyjádříme pomocí zápisu $|b|/||\vec{w}||$. Pro všechny vstupní vektory platí:

$$\vec{w} \cdot \vec{x} + b \geq +1 \text{ pro } y_\text{i} = +1, \tag{1.14}$$

$$\vec{w} \cdot \vec{x} + b \leq -1 \text{ pro } y_\text{i} = -1, \tag{1.15}$$

kde $\vec{x_i}$ představuje vstupní vektor, jenž s parametrem y_i udávajícím klasifikační třídu, vytváří množinu trénovacích dat $(\vec{x}_\text{i}, y_\text{i})$. Vztahy 1.14, 1.15 tedy vyjadřují samotné rozdělení vstupních dat do dvou tříd. Podpůrné vektory leží na nadrovině (v našem případě na přímce) p_1, respektive p_2, a jsou hraničním případem, pro který platí vztahy:

$$p_1 : \vec{w} \cdot \vec{x} + b = 1, \tag{1.16}$$

$$p_2 : \vec{w} \cdot \vec{x} + b = -1. \tag{1.17}$$

Tyto přímky jsou poté kolmé na normálový vektor $\vec{w}$ rozdělující nadroviny a vymezují hraniční pásmo. Od počátku souřadnic budou přímky vzdáleny podle vztahů:

$$\frac{|1 - b|}{||\vec{w}||} \text{ pro } p_1, \tag{1.18}$$

$$\frac{|-1 - b|}{||\vec{w}||} \text{ pro } p_2. \tag{1.19}$$

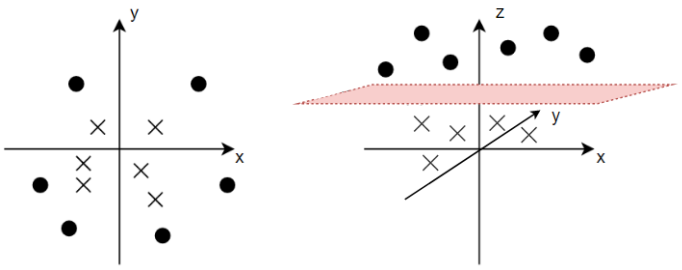
Jejich vzájemná vzdálenost poté bude rovna rozdílu jejich vzdáleností od počátku souřadnic, tedy $2/||\vec{w}||$. Pokud je soubor dat lineárně separabilní, nebude v hraničním pásmu, tedy mezi přímkami p_1 a p_2, ležet žádný vektor. Nejširší pásmo dosáhneme minimalizací $||\vec{w}||$ [28] při dodržení vztahů 1.14 a 1.15.

Do této doby jsme uvažovali nad problémy, které jsou lineárně separabilní. Při řešení složitějších problémů, které nelze jednoduše klasifikovat pomocí lineárních vztahů, se využívá následující myšlenky. Pomocí jádrové transformace se data z původního prostoru převedou do vyšší dimenze, ve které je již můžeme lineárně separovat. Provádíme tedy zobrazení Φ:

$$\Phi : \mathbf{R}^\text{d} \mapsto \mathbf{R}^\text{D}, \tag{1.20}$$

kde $\mathbf{R}^\text{d}$ je původní prostor a $\mathbf{R}^\text{D}$ prostor vyšší dimenze. Transformačních jader existuje celá řada, mezi nejvíce používané patří jádrové transformace Gaussova typu, polynomiální či sigmoidální [26]. Velmi jednoduchý příklad transformace můžeme ilustrovat pomocí následující problému, kdy uvažujeme lineárně neseparabilní datový

set bodů o souřadnicích x, y v počáteční rovině. Pomocí transformace $z = x^2 + y^2$, zavádíme nový popisný parametr pro data – ve své podstatě z vyjadřuje vzdálenost bodů od počátku souřadnic. Po transformaci do prostoru o vyšší dimenzi (v tomto případě $\mathbb{R}^3$) se problém stává lineárně separabilní.



Obr. 1.5: Lineárně separabilní případ po vhodné transformaci

Optimalizaci metody podpůrných vektorů můžeme dosáhnout pomocí dvou parametrů. První z nich, tzv. *gamma* parametr, určuje kolik bodů bude zahrnuto do konečného výpočtu. Nízké hodnoty tohoto parametru zajistí, že se bude počítat s větším počtem okolních bodů, vysoké hodnoty pak udávají, že do výpočtu budou zahrnuty jen body nejbližší. Na tento parametr můžeme nahlížet jako na inverzi poloměru kružnice, v jejíž vnitřní oblasti leží body ovlivňující výpočet modelu. Parametr označený písmenem C reprezentuje regulaci mezi chybou a komplexností celého výpočtu [29]. Udává tedy, do jaké míry je povolena nepřesná klasifikace vstupních dat.

Konvoluční neuronové sítě

V poslední řadě let dochází k rozmachu hlubokého učení. Určujícím elementem jsou konvoluční neuronové sítě (CNN – Convolution Neural Network), které dosahují výborných výsledků v oblasti počítačového vidění – rozpoznávání obrazů jako celku, detekci samotných objektů či popisu obrazové scény. Odpovídající výsledky konvolučních neuronových sítí je možné dosáhnout zejména pomocí vyšší dostupnosti trénovacích dat [30]. Pojem hlubokého učení (tzv. Deep Learning) je odvozen od velkého počtu skrytých vrstev (hloubky) neuronové sítě, tedy vrstev nacházejících se mezi vstupní a výstupní vrstvou. Popis základních pojmů z oblasti konvolučních neuronových sítí bude proveden v následujícím textu.

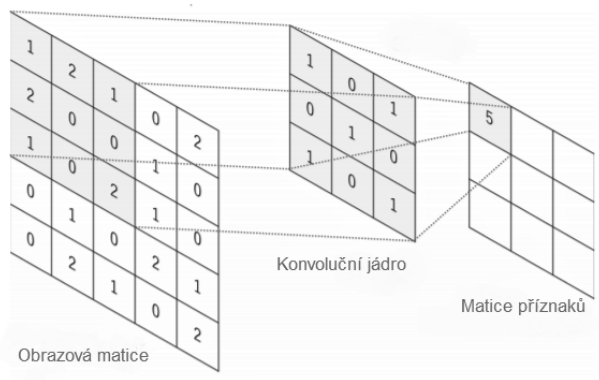
Základní jednotka konvoluční neuronové sítě je stejná jako u ostatních neuronových sítí – je reprezentována neuronem. Ve své podstatě je neuron definován přesným matematickým popisem, respektive přenosovou funkcí $y = f(z)$, která vstupy neuronu $x = [x_1, x_2, \ldots, x_n]$ převádí na výstup y, a kde z reprezentuje vnitřní potenciál

neuronu nastavený vahami pro jednotlivé vstupy. Samotné vstupy neuronu jsou realizovány pomocí vazeb mezi dalšími neurony – vzniká tak neuronová síť. Konvoluční neuronové sítě jsou tvořeny třemi typy vrstev [31].

Konvoluční vrstva realizuje extrakci příznaků vstupního obrazu pomocí konvoluce. Tato operace již byla zmíněna u metod segmentace využívajících hranové detektory, nicméně je vhodné ji znovu zmínit, jelikož tvoří kritickou část konvolučních neuronových sítí. Vstupní obraz je konvolován s filtrem, který v obraze hledá předurčené znaky, typicky hrany či barevné shluky. Konvoluční maska filtru je tedy postupně přikládána k obrazu a je tak vytvářena výsledná matice popisující míru daného příznaku v obrazu. Obecně diskrétní 2D konvoluci můžeme popsat vztahem [4]:

$$\mathbf{g}(i,k) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \mathbf{I}(i-m, k-n)\mathbf{h}(m,n), \tag{1.21}$$

kde $\mathbf{h}$ představuje konvoluční jádro a $\mathbf{I}$ diskrétní obraz. Velmi názorně je konvoluce zobrazena v animaci na webové stránce[2].



Obr. 1.6: Konvoluce obrazu a konvolučního jádra [31]

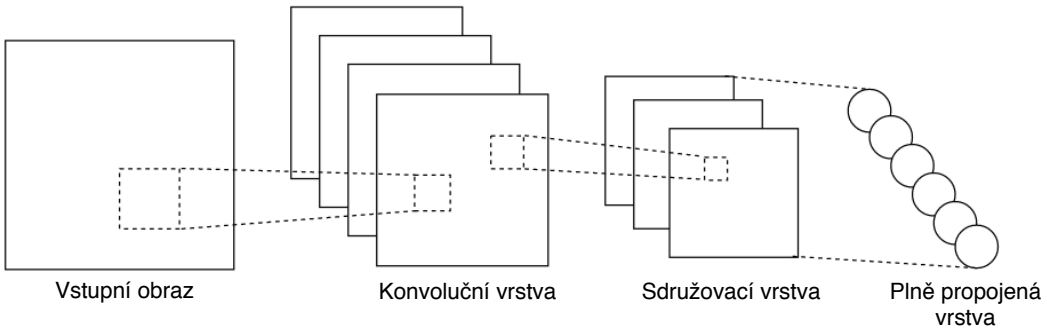
Další vrstvou je vrstva sdružovací, která zachovává jen důležité hodnoty a ostatní hodnoty eliminuje. Toto je velmi důležitý krok, neboť je poté snížen počet operací vyšších vrstev. V oblasti segmentace obrazu je často využíváno metod, které z dané oblasti matice vyberou jen nejvyšší hodnotu, popřípadě provádějí průměrování hodnot z daného okna. Dochází k redukci souboru hodnot na hodnotu jedinou [31], jinými slovy je realizováno podvzorkování obrazu.

Plně propojená vrstva pracuje s jednorozměrným vektorem hodnot. Tento vektor vstupuje do každého neuronu výstupní vrstvy, kde se již počítá konečný výstup celé konvoluční sítě. Plné propojení spočívá v propojení každého výstupu neuronu této vrstvy se vstupem neuronu dalšího. Plně propojené vrstvy jsou umístěny na

[2] http://cs231n.github.io/convolutional-networks/

závěr architektury a provádí samotnou klasifikaci. Platí, že poslední plně propojená vrstva obsahuje tolik neuronů, kolik existuje klasifikačních tříd. Možností uspořádání architektury existuje celé řada a je vždy nutné zvolit takovou architekturu, která odpovídá konečnému využití. Nicméně oproti jiným metodám neuronové sítě nabízí značnou flexibilitu díky schopnosti učit se na základě vstupních dat.

Po definování architektury sítě je důležité také zvolit další dva parametry. První z nich je ztrátová funkce (loss function), jejíž hodnoty se budou během trénování minimalizovat, svým způsobem představuje míru úspěšnosti klasifikace daného problému. Druhým je optimalizátor, který definuje aktualizaci sítě na základě výsledků ztrátové funkce. Je realizován konkrétní variantou stochastického gradientního sestupu [24]. Ve své jednodušší podstatě tento krok znamená najít minimum ztrátové funkce, respektive zjistit takovou kombinaci hodnot vah, kdy je docíleno nejmenší možné ztráty.

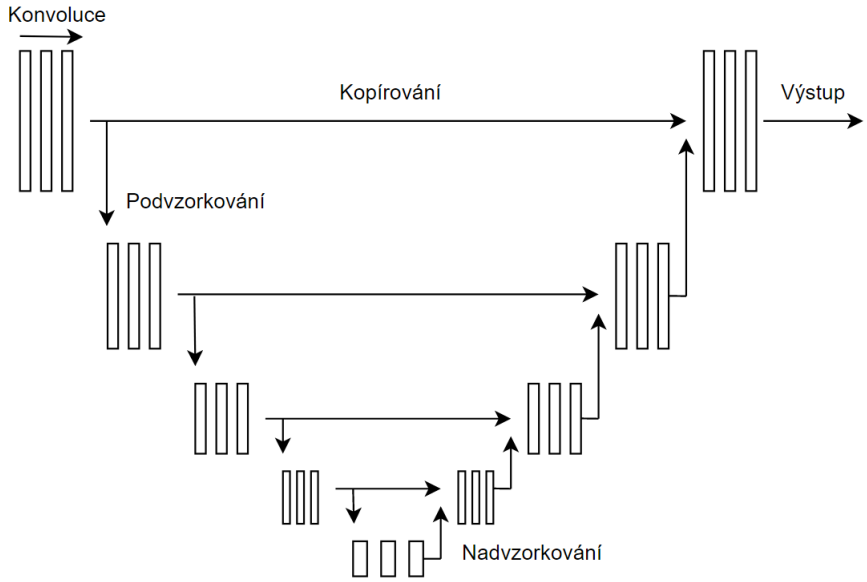


Obr. 1.7: Jednotlivé vrstvy konvoluční neuronové sítě

Architektura U-Net

Architektura U-Net byla přímo vytvořena pro segmentaci obrazů v medicínské oblasti. Byla poprvé publikována v článku Olafa Ronnebergera a jeho kolegů v roce 2015 [32]. Dosahuje kvalitních výsledků i při menším množství trénovacích dat a disponuje výpočetní rychlostí. Název vyplynul z uspořádání její vnitřní struktury, jež připomíná písmeno U. Levá strana architektury provádí podvzorkování a dochází tak ke snižování rozměrů matic, pravá strana naopak provádí nadvzorkování, které vede k expanzi rozměrů matic. Tyto operace jsou prováděni postupně a po stejných krocích. Nadvzorkování je provedeno nejen pomocí předem definovaných filtrů, ale také díky spojení odpovídajících vrstev levé a pravé části. Nedochází tak k pouhé interpolaci při zvyšování rozměrů matic, která by zanesla chybu do konečného výsledku segmentace [32]. V každém patře probíhá konvoluce za pomocí konvolučního

jádra o rozměrech 3×3, jež je následovaná lineární aktivační funkcí typu ReLU (Rectified Linear Units). Tato běžně používaná aktivační funkce v oblasti hlubokého učení vrací 0, pokud je jejím vstupem záporné číslo, v případě kladného vstupního čísla vrací právě toto číslo [33]. Konečným krokem je přiřazení do jednotlivých klasifikačních tříd.



Obr. 1.8: Jednotlivé vrstvy architektury U-Net

1.3 3D Slicer

Jedná se o volně dostupný (open source) software, jenž je určen pro práci s medicínskými obrazy a jejich následnou vizualizaci. 3D Slicer (dále jen Slicer) vznikl seskupením více samotných projektů, které se zabývaly vizualizací obrazu, navigací při operacích a koněčně také grafickým uživatelským rozhraním (Graphical User Interface – GUI). Prototyp softwaru byl prezentován již v roce 1999 v rámci diplomové práce Davida Geringa na Massachusettském technologickém institutu [34]. V současné době je Slicer vyvíjen nejen profesionály, ale i členy stále rozsáhlejší komunity.

1.3.1 Architektura

Z hlediska softwarového inženýrství je jádro Sliceru postaveno na architektuře MVC (Model–View–Controller). Tato architektura byla poprvé použita a popsána v sedmdesátých letech minulého století. Na původní zprávu poprvé zmiňující MVC je

možné nahlédnout online na odkazu[3]. Architektura je založena na třech základních pilířích, jež vykonávají specifickou funkci.

Model

Model reprezentuje data, s kterými lze manipulovat, v případě Sliceru se jedná o Medical Reality Markup Language (MRML), tedy značkovací jazyk používaný v medicínském prostředí, jenž definuje hierarchii dat a umožňuje samotný přístup k datům pomocí rozhraní (API). Knihovna MRML je také zodpovědná za koherenci interních dat v podobě MRML scény.

View

Jak už z názvu View (pohled) vypovídá, tento člen architektury má zá úkol vizualizaci, respektive prezentování dat uživateli. Prakticky tuto funkci vykonávají interní třídy Sliceru, jež se snaží o jednotnost vnitřního modelu a dat zobrazovaných uživateli. Poté díky aplikačnímu rámci (frameworku) *Qt*, pomocí kterého je vytvořeno samotné GUI, jsou data vizualizována.

Controller

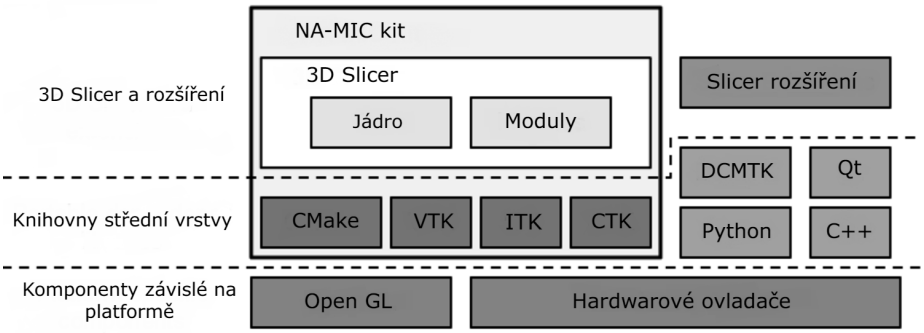
Controller (řadič) zodpovídá za reakce na události vytvořené uživatelem. V principu to znamená, že interně komunikuje s modelem či pohledem a zajišťuje jejich změny. Ztělesňuje samotnou logiku softwaru a zastává výpočetní funkci.

1.3.2 Knihovny

Jednotlivé úrovně softwarového systému Slicer využívají různých knihoven. Na nízké úrovni operují knihovny jako *OpenGL* či hardwarové grafické ovladače, jejichž poskytnutí zajišťuje sám operační systém. Další důležitou složkou je jazyk C++, který tvoří robustní kostru softwaru. Nelze ani opominout skriptovací jazyk Python, respektive jeho třetí verzi, jejíž specifika jsou výborně probrány v knize Marka Pilgrima [36] či přímo na webové stránce[4], a kterému bude v pozdější části práce věnováno větší pozornost. Ve shodné vrstvě se také vyskytují knihovny jako *VTK* (The

[3] http://heim.ifi.uio.no/trygver/1979/mvc-1/1979-05-MVC.pdf
[4] http://diveintopython3.py.cz/index.html

Visualization Toolkit) sloužící nejen pro 3D zobrazování scén, *ITK* (Insight Segmentation and Registration Toolkit) obstarávající nástroje pro zpracování obrazu či *CMake* podporující správu překladu softwaru [35].



Obr. 1.9: Architektura Sliceru [34]

1.3.3 Moduly a rozšíření

Poslední dvě části, jež nebyly v rámci architektury popsány, jsou moduly a externí rozšíření (extensions). Již s instalací Sliceru je distribuováno několik desítek základních modulů, které umožňují pokročilou práci s obrazovými daty. Další moduly vytvářené komunitou tohoto softwaru je možné získat pomocí záložky Extension Manager. Jednotlivé defaultní moduly jsou určeny k filtraci, registraci, segmentaci, vytvoření oblasti zájmu (Region of Interest – ROI) či ke správě DICOM souborů. Na stránce[5] jsou uvedeny možnosti vytváření rozšiřujících modulů. Celkově je možné modul implementovat třemi způsoby, o vhodnosti individuálního způsobu řešení rozhoduje typ vstupních parametrů.

Rozhraní příkazové řádky

Nejvíce limitovaným z hlediska vstupů je CLI modul (Command Line Interface – rozhraní příkazové řádky). Je omezen na vstupní a výstupní parametry bez průběžné uživatelské interakce. Uživatelské rozhraní (UI - User Interface) je k modulu generováno automaticky, popřípadě je ho možné měnit pomocí konfiguračního souboru XML (značkovací jazyk – Extensible Markup Language). Ve své podstatě nejvíce připomíná nativní Python skript bez žádných externích závislostí či složité hierarchie. V rámci programu Slicer je možné tento typ modulu volat z ostatních modulů a využít ho tak ke spuštění procesu, který běží na pozadí. Vývojář je poté schopný

[5] https://www.slicer.org/wiki/Documentation/Nightly/Developers/Modules

rozdělit uživatelské rozhraní běžící v prvním vláknu od části kódu vykonávající samotnou logiku programu běžící ve vláknu druhém.

Skriptovaný modul

Modul psaný v jazyku Python, tedy skriptovaný modul, umožňuje plnou komunikaci se zabudovanými knihovnami. V rámci Sliceru je navíc možné využívat konzoli (*Python Interactor*) pro zefektivnění práce. Pomocí knihovny *Qt* je pak možné vytvořit uživatelské prostředí. Jelikož Slicer od verze 4.11.0 podporuje Python 3 a také instalaci externích balíčků, je tento způsob implementace velmi relevantní.

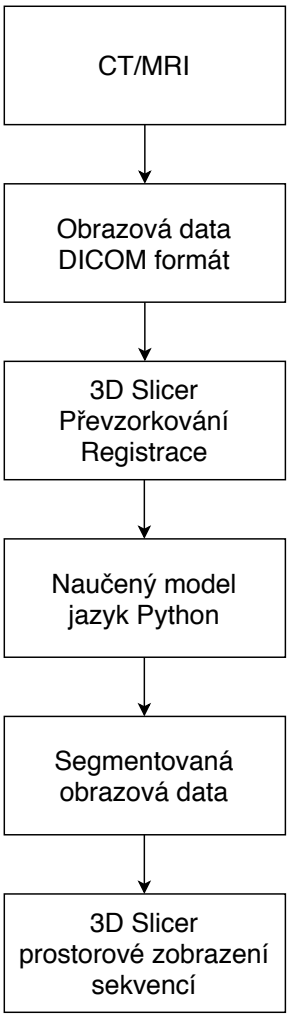
Vestavěný modul

Nejkomplexnější způsob tvorby modulu představuje implementace vestavěného modulu v jazyku C++. Jedná se o pokročilý postup, kdy si vývojář taktéž může nadefinovat vlastní grafické uživatelské rozhraní vyhovující specifické aplikaci. Moduly toho typu se kompilují současně se samotným Slicerem. Předností modulu je rychlost a možnost optimalizace pro náročné výpočty.

2 Experimentální část

2.1 Postup zpracování dat

Obrazová data ve formátu DICOM pořízena pomocí CT a MRI jsou nahrána do programu Slicer, kde probíhá převzorkování a samotná registrace obrazů. Tato problematika bude popsána v nadcházející kapitole. Těžištěm práce je poté segmentace obrazů pomocí modelu SVM, který bude naučen na trénovací množinou obrazových dat, jež byly ručně segmentovány. Jednotlivé masky segmentovaných řezů je dále možné uspořádat do výsledné trojrozměrné masky a získat tak prostorovou reprezentaci obratle, popřípadě celé páteře. Postup zpracování obrazových dat může být popsán diagramem uvedeným níže.



Obr. 2.1: Diagram znázorňující postup zpracování dat

2.2 Obrazová databáze

Jak už bylo zmíněno v úvodu, pro účely této práce byly poskytnuty obrazy páteře s okolním svalstvem a také bez svalstva. Prvotním krokem je výběr vhodné sekvence, která poskytuje dostatečný kontrast mezi kostí a pozadím, a vůči které budou ostatní sekvence registrovány. Po segmentaci této vhodné sekvence vznikne binární maska, která bude reprezentovat nejpřesnější možný výsledek. S výslednou maskou pak budou porovnány výsledky segmentace u obrazů, které neposkytují tak dobrý kontrast – jedná se typicky o obrazy pořízené MRI. Z poskytnutých obrazových dat byly vybrány sekvence, jejichž parametry jsou uvedeny v následujících tabulkách. Měření bylo provedeno pro jeden obratel separovaný z obrazu.

Tab. 2.1: Přehled vybraných sekvencí pro následnou registraci a segmentaci

Základní informace			Intenzita pixelů				
Sekvence	Rozlišení	Řezy	Min	Max	Průměr	Odchylka	Medián
`CT bez tkáně (A)`	512×512	812	-1024	1548	-735	402	-929
`CT bez tkáně (B)`	512×512	812	-1024	3043	-702	543	-983
`MRI T1 bez tkáně`	1024×1024	26	241	3222	2245	402	2330
`MRI T2 bez tkáně`	512×512	168	24	2190	1354	509	1511
`MRI T1 s tkání`	1008×1008	26	26	2469	1104	441	1077
`MRI T2 s tkání`	784×784	41	10	2243	703	536	485

Tab. 2.2: Přehled jasových hodnot jednotlivých sekvencí.

Sekvence	$\mu_{\text{pozadí}}$	μ_{obratel}	Rozdíl	$\sigma_{\text{pozadí}}$	SNR	CNR
`CT bez tkáně (A)`	-989	-358	631	10,2	65,3	61,9
`CT bez tkáně (B)`	-976	-417	559	29,4	20,6	19,0
`MRI T1 bez tkáně`	2595	2098	497	128,4	16,3	3,9
`MRI T2 bez tkáně`	1783	1059	724	91,5	11,6	7,9
`MRI T1 s tkání`	949	1465	516	104,4	14,0	9,9
`MRI T2 s tkání`	788	376	412	106,9	3,5	6,4

Tabulka 2.2 vyjadřuje průměr jasových hodnot v okolí obratle a průměr jasových hodnot uvnitř obratle, rozdílem průměrů se poté snaží demonstrovat jejich vzájemný odstup a vyjádřit tak míru separovatelnosti obratle od pozadí. Obdobně pa-

rametr SNR (signal-to-noise ratio) se snaží popsat kvalitu obrazu pomocí vztahu:

$$SNR = \frac{\mu_o}{\sigma_p}, \tag{2.1}$$

kde μ_o udává průměr intenzity uvnitř obratle a σ_p pak směrodatnou odchylku jasových hodnot pozadí. Parametr CNR (contrast-to-noise ratio) dále popisuje kvalitu obrazu pomocí poměru:

$$CNR = \frac{|\mu_p - \mu_o|}{\sigma_p}, \tag{2.2}$$

kde μ_p a μ_o je průměr intenzity mimo, respektive uvnitř obratle, σ_p je již zmíněná směrodatná odchylka jasových hodnot pozadí. Jako referenční sekvence, vůči které budou ostatní sekvence registrovány a jejíž maska bude maskou výslednou, byla vybrána sekvence bez tkáně pořízená pomocí CT (A), jež poskytuje dostatečný kontrast, a kde kontury samotných obratlů jsou zřetelně odděleny od pozadí. Po tomto výběru se již může začít s koregistrací.

2.2.1 Registrace obrazů

Pro úspěšnou registraci obrazů je nutné se uživatelsky seznámit s programem Slicer do větší hloubky. Prvotním krokem je nahrání vybraných sekvencí do samotného Sliceru a následné převzorkování ostatních obrazů tak, aby jejich rozlišení odpovídalo obrazu referenčnímu, tj. CT obrazu s rozměry 512×512. Po převzorkování může nastat samotná registrace – zde se však dostáváme k prvnímu významnému problému v rámci řešení práce. Poskytnuté obrazy na sebe prostorově nelícují, tedy jednotlivé sekvence v rámci CT a MRI jsou prostorově různě orientovány. Tato chyba je způsobena absencí informace o prostorové orientaci sekvence v hlavičce DICOM souboru. Konkrétně se jedná o atribut (0020, 0032) *Image Position (Patient).* Sám Slicer při nahrávání sekvencí zobrazí upozornění na tento problém. Informace o prostorovém umístění je zřejmě chybějící z důvodu povahy snímaného objektu – nejedná se o pacienta, nýbrž o torzo. Chyba je způsobena také manipulací s torzem při dvoumodálním snímání. Pro uskutečnění dalších kroků v rámci registrace je nejprve nutné obrazy přesunout tak, aby se alespoň z části překrývaly. Tato úvodní komplikace velmi zpozdila řešení samotné segmentace.

Problém spočívá v definování vhodného obrazového prostoru. Bylo prozkoumáno několik možností, jak dojít k řešení situace. Tou z počátku nejrelevantnější cestou byla cesta následující – Slicer ukládá obrazové sekvence ve formátu nrrd (Nearly Raw Raster Data). Prvotním krokem tedy bylo načtení dokumentace k tomuto formátu, kde je zmíněno, že k hlavičce, která obsahuje parametry o obrazovém souboru, lze

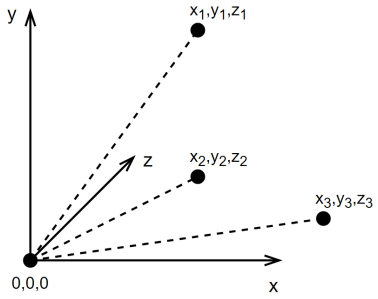
snadno přistoupit pomocí textového editoru. Nachází se v ní čtyři důležité parametry udržující prostorové informace o obrazové sekvenci:

- *space*,
- *sizes*,
- *space origin*,
- *space directions*.

Parametr *space* definuje orientaci rastru vůči pacientovi, v našem případě je v parametru udána orientace *left-posterior-superior (LPS)*. Tato orientace je všech zkoumaných sekvencích shodná. Parametr *sizes* reprezentuje údaj o rozměrech obrazu a počtu řezů. Parametr *space origin* poté udává polohu počátečního obrazu sekvence, respektive jejího prostředního pixelu. Rozhodujícím parametrem pro náš problém je pak parametr *space directions*, který se zapisuje pomocí prostorových souřadnic tří vektorů ve formátu:

$$(x_1, y_1, z_1)(x_2, y_2, z_2)(x_3, y_3, z_3), \tag{2.3}$$

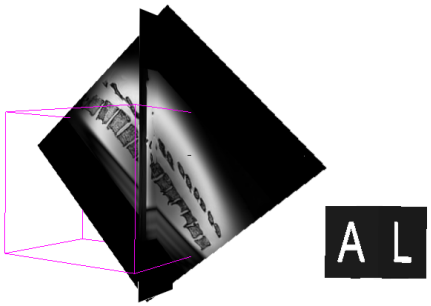
kde jednotlivé konstanty x_i, y_i, z_i vyjadřují směr vektorů v osách x, y, z.



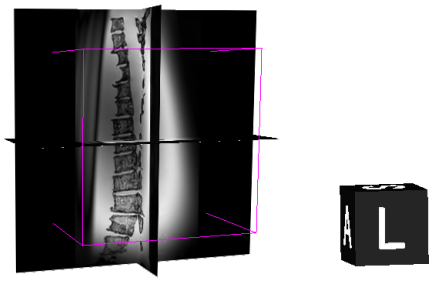
Obr. 2.2: Vektory společně s počátkem souřadnic definují prostor

Ve své podstatě konstanty neudávají jen směr, ale také vzdálenost sousedních voxelů v daném směru. Společně s počátkem souřadnic $(0, 0, 0)$ tyto vektory pak definují prostor obrazu. U sekvencí pořízených pomocí MRI byly hodnoty vektorů postupně měněny tak, aby se docílilo prostorového srovnání s referenční sekvencí pořízené pomocí CT. Touto metodou se však podařilo prostorově srovnat jen jednu sekvenci. Navíc nelze najít univerzální transformaci, která by šla aplikovat na další sekvence, jelikož nabývají jiných rozměrů a liší se v počtu řezů. V rámci metody bylo vyzkoušeno různých postupů, a to postavených i na základě matematického usuzování, bohužel žádný postup nevedl k odpovídajícímu výsledku elegantní cestou.

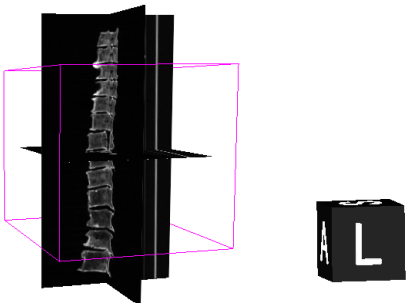
Metoda, pomocí které nakonec bylo dosáhnuto prostorového srovnání obrazů, byla čistě vizuální. Slicer dokáže zobrazovat sekvence v 3D prostoru, zároveň také umožní



(a) Původní MRI obraz



(b) MRI obraz po transformaci v prostoru



(c) Referenční CT obraz

Obr. 2.3: Transformace MRI obrazu k referenčnímu CT obrazu

vizualizaci více sekvencí najednou. Poté je velmi jednoduché si zobrazit referenční CT sekvenci v prostoru společně s MRI sekvencí. Pomocí modulu *Volumes*, konkrétně tlačítko *Center Volume*, lze jednotlivou sekvenci umístit do počátku souřadnic vizualizačního okna. Pokud tuto operaci provedeme s oběma sekvencemi, začnou se sekvence prostorově překrývat.

Tab. 2.3: Srovnání parametrů MRI obrazu po transformaci

Sekvence	*space origin*	*space directions*
`Původní MRI obraz`	(-166,4; 8,7; 54,6)	(0,45; 0; 0,39)(0,39; 0; 0,45)(0; 0,65; 0)
`Transformovaný MRI obraz`	(-45,4; -145,3; -157,3)	(0; 0,60 ;0)(0; 0; 0,60)(0,60; 0; 0)

Nicméně je nutný ještě další krok – se sekvencí MRI musíme rotovat v prostoru tak dlouho, dokud dílčí roviny (transverzální, sagitální, frontální) nebudou odpovídat dílčím rovinám referenční sekvence CT. To je možné realizovat pomocí transformace v modulu *Transforms*. Pro maximalizaci lícování obrazů je také nutné použít posunutí v jednotlivých osách.
Dalším problémem při zpracování dat je nemožnost uskutečnění globální registrace. Jelikož s páteří bylo při přenášení manipulováno, tak se u jednotlivých pořizovacích

(a) CT obraz bez svalu

(b) MRI obraz se svalem

Obr. 2.4: Porovnání obrazů dvou modalit

modalit poměrně liší její zakřivení. Ještě markantnější rozdíl zakřivení lze spatřit při porovnání páteře se svalem a bez svalu, viz obrázek 2.4. Registrace proto bude muset proběhnout po částech, tedy po skupinách obratlů, v některých případech pak i po jednotlivých obratlích. Z hlediska následné segmentace pomocí metod strojového učení se nejedná o nijak zvláštní komplikaci. Model bude naučen segmentování na úrovni dílčích obratlů, nicméně tyto obratle bude schopen segmentovat i v obrazech s celou páteří. Potíž je jen s pracností a celkovým počtem registrací – z každého obrazu budou muset být vybrány skupiny obratlů a následně registrovány se skupinou obratlů z referenčního obrazu. Registraci ve Sliceru zařizuje modul *General Registration (BRAINS)*. Tento modul nabízí několik možných typů registrace. V nejzákladnější formě je schopen provést rigidní transformace, tedy posunutí v prostoru. Pro pokročilejší operace pak nabízí změnu měřítka, zkosení a transformaci typu spline, jejíž použití je pro náš případ stěžejní – registrované obratle musí být deformovány tak, aby se dosáhlo maximálního srovnání vůči obratlům referenčním. V rámci modulu je nabízena řada pokročilých optimalizačních parametrů pro registraci, které uživatel může měnit. Mezi nejdůležitější z nich patří:

- *Max Iterations* vyjadřuje maximální počet iterací před ukončením registrace.
- *Relaxation Factor* udává poměr mezi přesností a rychlostí registrace. Pohybuje se v rozmezí $0 - 1$, kde vyšší hodnoty značí větší kompenzaci původních prostorových nesrovnalostí obrazů, avšak zvyšují čas potřebný pro registraci.
- *Reproportion Scale* určuje míru změny měřítka.
- *Skew Scale* popisuje míru zkosení.

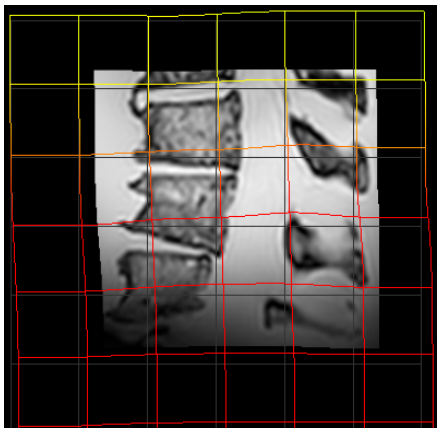
Pro registraci byla v modulu zvolena registrační metoda *BSpline*, jedná se o me-

todu výpočetně nejnáročnější, nicméně v našem případě je její použití nezbytné, jak bylo zmíněno výše. V následující tabulce jsou uvedeny další parametry registrační metody, jež byly použity.

Tab. 2.4: Přehled registračních parametrů metody *BSpline*

Parametr	Hodnota	Popis
Percentage of Samples	0,002	Počet voxelů použitých k registraci
B-Spline Grid Size	14, 10, 12	Dělení mřížky v jednotlivých osách
Histogram bin count	50	Počet jasových úrovní histogramu
Histogram match point count	10	Počet úrovní pro následné srovnání metrikou
Cost metric	MMI[1]	Metrika definována pomocí ztrátové funkce

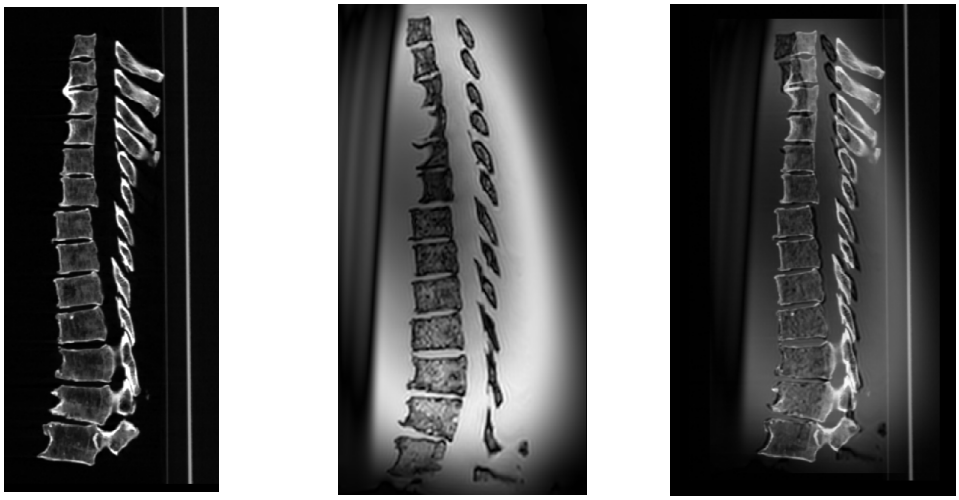
Samotnou transformaci můžeme zobrazit pomocí mřížky, která představuje prostorovou deformaci obrazu. Transformační mřížka registrované skupiny obratlů je zobrazena na následujícím obrázku. Je zde znázorněna také nezměněná původní čtvercová mřížka.



Obr. 2.5: Mřížka vyjadřující deformaci.

Pro účely práci byl také použit modul *General Registration (Elastix)*, jenž vyžaduje instalaci pomocí správce rozšíření.Tento modul v některých případech poskytuje lepší výsledky něž předchozí model. Dokumentace o tomto modulu neposkytuje značný počet informací, nicméně uživatelova interakce je omezena na minimum – je zvolen jen referenční obraz a obraz určený k registraci. Nejlepší výsledek globální registrace, kterého bylo dosaženo v rámci řešení práce, je uveden na obrázku 2.6.

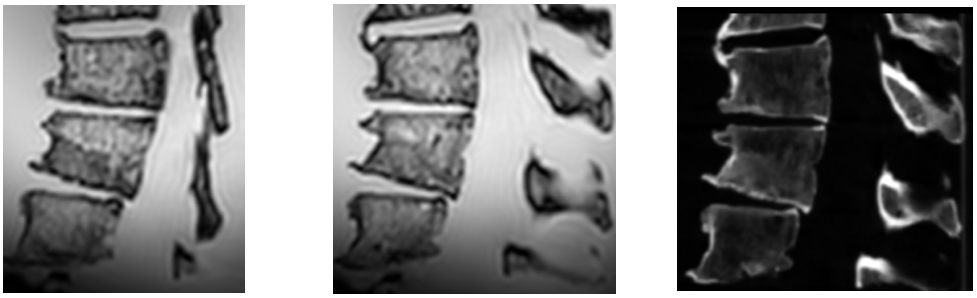
[1]V tomto případě se jedná o *Mattes Mutual Information* pracující na základě pravděpodobnosti popisující podobnost obrazů pomocí jejich histogramů



(a) CT obraz (b) MRI obraz (c) Překrytí obrazů

Obr. 2.6: Výsledek globální registrace pomocí splajnů

Značné překrytí je viditelné ve střední části obrazu, nicméně ve spodní a zejména v horní části je registrace značně nedostatečná. Úspěšná lokální registrace, tedy registrace jen samotné skupiny obratlů, je pak znázorněna na následujícím obrázku. Můžeme vidět, že po transformaci a prostorové deformaci obrazu MRI, je dosaženo dostatečného překrytí s referenčním obrazem CT.



(a) Původní MRI obraz (b) Registrovaný obraz (c) Referenční CT obraz

Obr. 2.7: Registrace jednotlivé skupiny obratlů

2.2.2 Export a příprava dat

Po úspěšné registraci se obrazy z prostředí Slicer mohou posléze vyexportovat jako jednotlivých řezy v DICOM formátu. Druhou možností je pak registrované obrazy uložit ve formátu nrrd. Před samotným exportem je vhodné obrazy orientovat do sagitální roviny, jelikož v této rovině řezy poskytují nejkvalitnější rozlišení. Toho lze u příslušné obrazové sekvence dosáhnout pomocí modulu *Orient Scalar Volume*. Důvod exportu dat je následující - segmentaci obrazů lze samozřejmě realizovat v rámci programu Slicer právě pomocí rozšiřujícího modulu SVM, který je jádrem a hlavním výstupem této bakalářská práce. Nicméně pro účely prvotního vývoje byla nejvíce optimální variantou implementace základní funkcionalitu jen v podobě nativních Python skriptů. Vývojář tak není zatížen dalším systémem a může se efektivněji věnovat samotného vývoji hlavní funkcionality. Po tomto vývoji bude logika programu propojena s uživatelským prostředím a vznikne tak plnohodnotný rozšiřující modul platformy Slicer pro segmentaci obrazů pomocí modelu SVM.

Po exportu je tedy nutné provést na vybraných obratlích ruční segmentaci. Tuto segmentaci je možné provádět v programu ImageJ. Masky je možné z programu ImageJ vyexportovat jako osmi bitové obrazy ve formátu png (Portable Network Graphics) a připravit je tak pro další zpracování. Následujícím krokem je příprava obrazových řezů a masek do podoby, jež bude klasifikátor SVM schopný zpracovat. Tento úkol zajišťuje skript s názvem *features.py*. Skript provádí kompresi a normalizaci obrazových dat, ekvalizaci histogramu, extrakci příznaků a v neposlední řadě upraví masky původně vyexportované z programu ImageJ, tak aby nabývaly binárních hodnot (1 – obratel, 0 – pozadí). Příznaky jsou extrahovány pro každý pixel, podrobnosti o této extrakci jsou uvedeny v následující části textu. Velmi důležitý je již zmíněný tvar příznakového vektoru, který je určen samotným klasifikátorem. V syntaxi jazyka Python vypadá příznakový vektor (lépe řečeno 2D pole) a maska rozvinutá do vektoru následovně:

```
features =[[x, x, x], [y, y, y], [z, z, z], ...]
labels = [0, 0, 1, ...]
```

Skupině příznaků `[x, x, x]` je na odpovídajícím indexu přidělena třída (*label*) s hodnotou 0, která tedy určuje, že se jedná o příznaky pixelu patřícího k pozadí. Výpis části skriptu realizující vytvoření příznakového vektoru je uveden níže. Skript dále data ukládá do datové struktury (*numpy array*) realizované pomocí knihovny *numpy*, která je standardem pro práci s číselnými daty v jazyce Python. Tento krok umožní pohodlný přístup k datům při následném trénování modelu pomocí metody `load_features()`.

Výpis 2.1: Python skript realizující přípravu příznakového vektoru

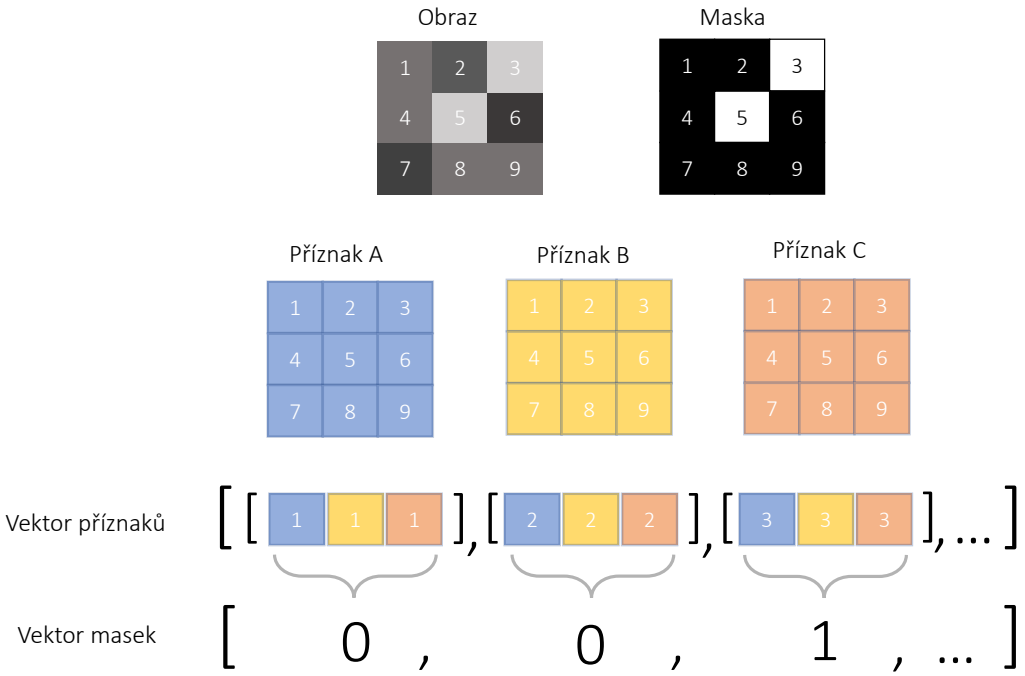
```
    features = []
    for image_path in sorted(os.listdir(data_path)):
        image = pydicom.dcmread(os.path.join(data_path, image_path)).pixel_array
        image = to_signed(image)
        image = normalize(image)
        image = equalize_histogram(image)
        feature_dictionary = get_feature_dictionary(image)
        # For each pixel
        for i in range(image.size):
            feature_row = []
            for feature in feature_dictionary.values():
                if isinstance(feature[0], np.ndarray):
                    feature_row.extend(feature[i])
                else:
                    feature_row.append(feature[i])
            features.append(feature_row)
```

Slovník *feature_dictionary* představuje datovou strukturu pro ukládání příznaků aktuálního obrazu a mapuje jednotlivé metody zajišťující extrakci příznaků k odpovídajícím názvům.

Výpis 2.2: Slovník příznaků

```
def get_feature_dictionary(image):
    return {
        'pixel_value': np.array(image).ravel(),
        'mean': mean(image),
        'variance': variance(image),
        'gaussian_filter': gaussian_filtering(image),
        'median_filter': median_filtering(image),
        'sobel_operator': sobel_operator(image),
        'gradient_matrix': count_gradient(image),
        'laplacian': laplacian(image)
    }
```

Pro názornost je na obrázku 2.8 uveden postup změny tvaru jednotlivých příznakových matic do výsledného příznakového vektoru. Pozornost při přípravě dat musí být také kladena na již zmíněnou normalizaci a standardizaci dat. Pokud data nejsou převedena do odpovídající rozsahu 0 - 1, velmi se prodlužuje čas trénování klasifikátoru. Toto platí pro každý příznak - pokud by jeden z příznaků ve srovnání s ostatními nabýval velkých hodnot, stal by se dominantním a následná klasifikace by byla zkreslená. V dokumentaci knihovny *scikit-learn*, která poskytuje implementaci modulu SVM, jež se bude v práci využívat, je s důrazem uvedeno, že algoritmus modelu SVM optimálně klasifikuje právě při předložení normalizovaných či standardizovaných dat.



Obr. 2.8: Tvorba příznakového vektoru

2.3 Tvorba modelu

Jak už bylo zmíněno v minulé kapitole, pro tvorbu modelu SVM realizujícího klasifikaci dat byla využita knihovna *scikit-learn*, která nabízí implementaci SVM v podobě klasifikátoru SVC. Klasifikátoru lze předat poměrně širokou škálu parametrů, ty nejdůležitější jsou uvedeny v tabulce 2.6.

Kód implementující klasifikátor se nachází v souboru *classifier.py*. Po samotné klasifikaci dat je problematickou částí zpětná změna tvaru predikovaných masek z formátu vektoru do dvojrozměrného obrazu, tak aby výška a šířka obrazu s maskou odpovídala rozměrům obrazům, pro které byly masky predikovány. Aby bylo možné masky takto transformovat, je nutné mít uloženou informaci o původních rozměrech. Toto v kódu zajišťuje datová struktura `features_info`, která je tvořena jednotlivými slovníky, jež popisují přiřazený obraz – ukládají informaci o počtu pixelů, výšce a šířce původního obrazu.

```
info = {
        'num_of_pixels': image.size,
        'height': image.shape[0],
        'width': image.shape[1],
    }
```

Tab. 2.5: Parametry klasifikátoru SVC

Parametr	Popis	Používaná hodnota
`max_iter`	Maximální počet iterací	10000
`tol`	Tolerance - regulace ukončení	1
`cache_size`	Velikost kešování v MB	500
`kernel`	Typ transformačního jádra	'rbf'
`c`	Parametr C	Závislá na datech
`gamma`	Parametr *gamma*	Závislá na datech
`max_samples`	Parametr používaný k rozdělení vstupních dat	1/16
`n_jobs`	Počet souběžné běžících operací	-1 (využití všech jader)
`verbose`	Míra výpisu logovacích zpráv	Při běhu na pozadí je doporučeno vypnout

Nejméně přehlednou částí je pak kód uvedený v následujícím výpisu 2.3. Jedná se právě o transformaci predikované masky v podobě vektoru do dvourozměrného obrazu. Pro aktuální iteraci je zvolen příslušný slovník, dále je zjištěno, kolik pixelů obsahuje jemu přiřazený obraz. Tento počet pixelů je poté vybrán z vektoru `predicted_masks`, jenž představuje pole, do kterého je po predikování uložen výsledek. Dále je pomocí slovníku zjištěna výška a šířka původního obrazu a je provedena změna tvaru. Poté jsou aplikovány matematické holomorfní operace z knihovny *scipy.ndimage*, jež jsou schopné odstranit nespojitosti či šum ve výsledných maskách. Predikovaná maska je uložena a pokračuje se další iterací.

Výpis 2.3: Zpětná změna tvaru masky

```
1     for i in range(num_of_images - NUM_OF_VALIDATION_IMAGES, num_of_images):
2         cur_mask_pixs = features_info[i]['num_of_pixels']
3         mask = predicted_masks[prev_mask_pixels:prev_mask_pixels + cur_mask_pixels]
4         mask = mask.reshape(features_info[i]['height'], features_info[i]['width'])
5         mask = ndimage.binary_opening(mask)
6         mask = ndimage.binary_closing(mask)
7         prev_mask_pixels = cur_mask_pixels
8         masks_predicted += 1
9         plt.imsave(os.path.join('predictions/',
10        'predicted_mask_' + str(masks_predicted) + '.png'), mask, cmap='gray')
```

Skript *classifier.py* nabízí tři režimy spuštění. Konkrétní aplikací těchto režimů se budou zabývat následující kapitoly.

Tab. 2.6: Režimy spuštění skriptu

Režimy spuštění	Popis
`'DEFAULT_TRAINING'`	Představuje jednorázové natrénování klasifikátoru, použije se dvojice parametrů C a *gamma*
`'GRID_SEARCH'`	Prohlédá prostor zadaných parametrů, provede křížovou validaci pro každou dvojici parametrů C a *gamma*
`'FROM_SAVED_CLASSIFIER'`	Umožňuje vyvolat již natrénovaný klasifikátor

2.3.1 Práce na serveru

Pro řešení práce bylo autorovi umožněno se vzdáleně připojit k fakultnímu serveru. To přineslo značné zrychlení práce, jelikož bylo možné skripty pouštět vzdáleně na serveru, což zejména při delších trénovacích cyklech v režimu *'GRID_SEARCH'* bylo velmi nápomocné. Zajímavou části práce bylo vykonávání následujících operací na serveru v prostředí Linux. Prvotním krokem bylo vytvoření virtuálního Python prostředí pro sandboxové oddělení od lokálního prostředí. Virtuální prostředí takto nabídne možnost instalace balíčků, aniž by došlo k jakémukoliv křížení závislostí mezi zbytkem systému. Prostředí se vytváří velmi jednoduše – po instalaci samotné služby *virtualenv*, lze následujícím příkazem provést inicializaci nového virtuálního prostředí:

```
pip3 install virtualenv
python3 -m venv <název_prostředí>
```

Poté je nutné virtuální prostředí aktivovat pomocí:

```
source <název_prostředí>/bin/activate}
```

Nyní je možné pokročit k instalaci balíčků. Značnou výhodou je možnost vytvoření seznamu balíčku, které je nutné mít nainstalovány ke korektními spuštění daného projektu, prostředí se tak stává přenositelným. Seznam lze vytvořit a následně vyvolat těmito příkazy:

```
pip3 freeze > <název_seznamu>.txt
pip3 install -r <název_seznamu>.txt
```

Pro skripty běžící na serveru je důležité, že jejich běh nebude ovlivněn odhlášením uživatele. Tuto možnost nabízí příkaz `nohup`, který pokračuje ve svém výkonu

i po odhlášení uživatele. V kontextu řešení práce byl příkaz použit současně s přesměřováním výstupu do souboru, který je konstantně aktualizován výpisy běžícího skriptu.

```
nohup python3 -u ./classifier.py ct > SVM_output.log &
```

Pro dávkové zpracování obrazů všech modalit a jejich segmentaci byl napsán skript *run_all.sh*, při jehož spuštění v kombinaci s příkazem `nohup` se na serveru provedou dané úkoly v pozadí. Výsledky těchto operací jsou uvedeny v závěrečné kapitole.

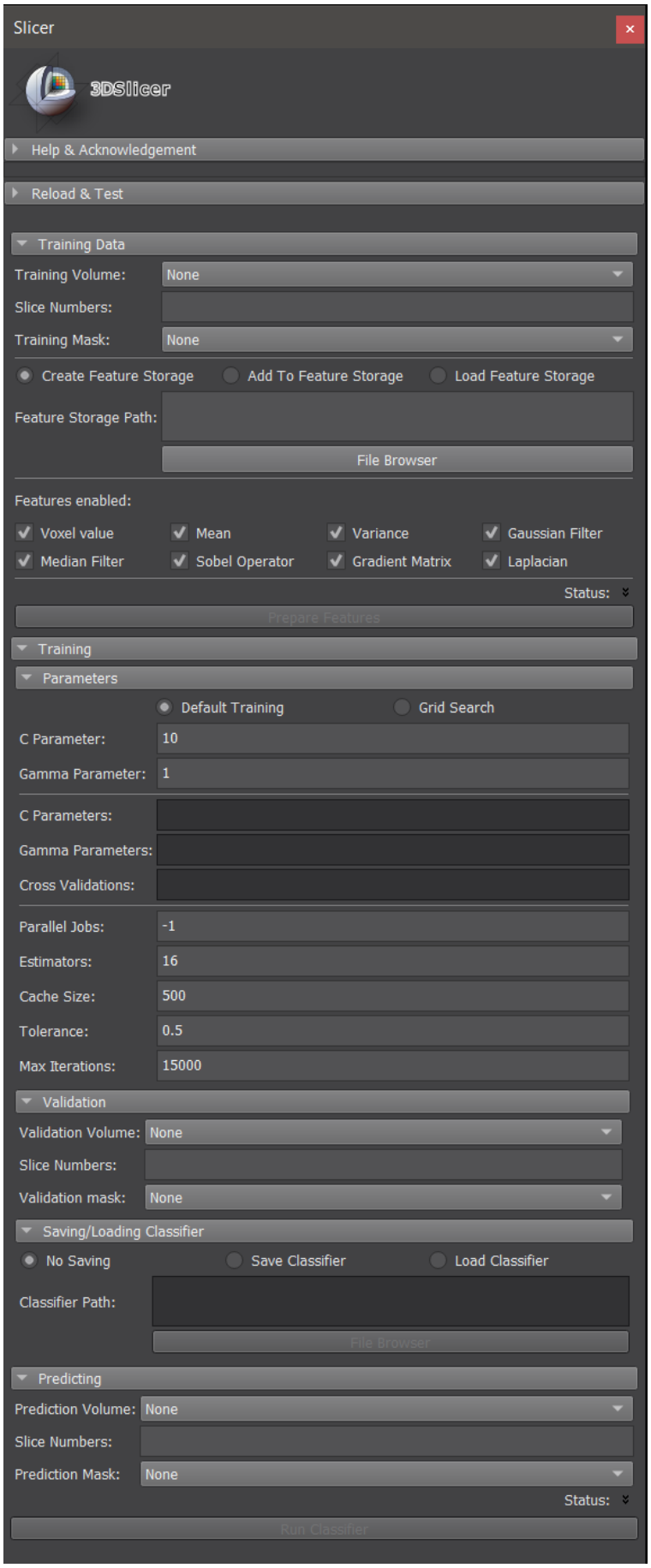
Výpis 2.4: Ukázka části skriptu

```
1 #!/bin/bash
2 printf "CT classification running...\n"
3 python3 -u features.py ct >> SVM_ct_out.log && \
4 python3 -u classifier.py ct  >> SVM_ct_out.log
5 wait
6 printf "MRI T1 BONE classification running...\n"
7 python3 -u features.py mri_t1_bone >> SVM_t1_b_out.log && \
8 python3 -u classifier.py mri_t1_bone >> SVM_t1_b_out.log
```

2.3.2 Implementace modulu v programu Slicer

Jak bylo zmíněno v předcházející části textu, logika samotného modulu do jisté míry odpovídá skriptům, jež byly spouštěny na serveru. Modul je nicméně složitější a nabízí uživateli širší možnosti. Snaží se současně poskytnout přehledné uživatelské prostředí s několika rozšířeními oproti stávajícím skriptům. Grafické uživatelské prostředí modulu je zobrazeno na obrázku 2.9.

Modul je rozdělen do tří částí, které korespondují s průběhem operací během zpracování obrazových dat. Sekce *Training Data* umožnuje uživateli výběr trénovací sekvence obrazových dat (ve Sliceru nazývaných jako *Scalar Volume Node*) a korespondující sekvence s maskami (*Label Map Volume Node* nebo *Scalar Volume Node*). Dále je nutné zvolit vybrané řezy. Korektní formáty zadávání čísel řezů jsou následující – buď lze použít čísla řezů oddělená čárkou (99, 100, 101) nebo rovnou zadání rozsahu (99 - 101). Dále uživatel může vytvořit vektor příznaků, přidat příznaky k již existujícímu vektoru či jen načíst vektor pro účely dalšího zpracování. Po zvolení cesty k souboru, je poté na výběru uživatele, které příznaky použije pro množinu obrazů, jež se chystá segmentovat. Tento systém ukládání příznakového vektoru do souboru je relevantní z hlediska časové nenáročnosti, při opětovném spuštění Sliceru nemusí uživatel vytvářet nový příznakový vektor, ale stačí načíst již existující. Po kliknutí na *Prepare Features* je spuštěn skript, jenž běží na pozadí a realizuje extrakci příznaků, svou funkcí odpovídá skriptu *features.py* z předešlých kapitol. Po doběhnutí operace extrakce příznaků je uživateli zobrazena zpráva se



Obr. 2.9: Uživatelské rozhraní modulu

základními informacemi o výsledku přípravy příznakového vektoru. Další sekcí je sekce *Training* sloužící pro zadávání parametrů klasifikátoru a výběru režimu trénovací fáze. Validním formátem pro zadávání parametrů *gamma* a *C* při zvoleném režimu *Grid Search* je zadávání jednotlivých parametrů v podobě hodnot, jež jsou odděleny čárkou. Dále je zde nabízena možnost zvolení validačního setu, tedy obrazové sekvence s již připravenými maskami tak, aby bylo možné validovat a určit úspěšnost klasifikátoru. Uživatel si může natrénovaný model uložit či opět vyvolat. Sekce *Predicting* umožňuje uživateli vybrat obrazovou sekvenci dat, pro kterou bude prováděna predikce, zároveň zde má možnost vytvořit prázdnou sekvenci masek, do které bude posléze propsána predikovaná maska. Po kliknutí na tlačítko *Run Classifier* je spuštěn běh klasifikátoru, který poté realizuje segmentaci obrazových dat.

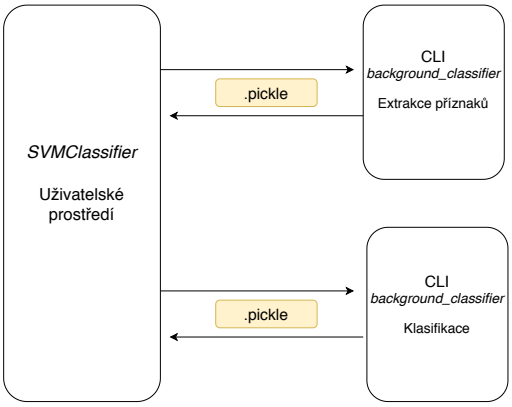
Důležitým bodem je jednoduchý popis architektury modulu, kód grafického uživatelské prostředí je umístěn v modulu *SVMClassifier.py*. Jedná se o skriptovaný modul, samotná logika, tedy příprava příznaků a klasifikace obrazových dat, je však implementovaná v modulu *background_classifier.py* - jedná se o CLI modul. Důvod oddělení uživatelského prostředí a výpočetní logiky je jednoduchý - z modulu *SVMClassifier.py* můžeme zavolat *background_classifier.py*, výpočet klasifikace bude přesunut do separátního vlákna a nebude tak docházet k zamrznutí uživatelského prostředí. Komunikace mezi moduly probíhá pomocí *pickle* souborů, ve své podstatě se jedná se o serializované objekty (konkrétně slovníky), které jsou datovou strukturou pro uložení potřebných dat. Při zjednodušení by konkrétní slovník, ve kterém by byla uložena data, mohl vypadat následovně:

```
feature_storage = {
    'training_data': get_training_volume(),
    'training_mask': get_training_mask(),
    'enabled_features': get_enabled_features(), ...
}
```

Tento slovník by byl naplněn daty z uživatelského prostředí, poté serializován do *pickle* souboru, předán do modulu *background_classifier.py*, zde by byl následně deserializován, proběhla by extrakce příznaků a připravený vektor příznaků by byl přidán do slovníku, jenž by byl opět serializován, a komunikace by probíhala opačným směrem. Ve skutečnosti si skripty mezi sebou vyměňují jen adresářní cestu k těmto souborům. Samotné volání CLI modulu je možné uskutečnit metodou `slicer.cli.run()`. Metoda vyjadřuje skutečnost, že volaný modul poběží nezávisle a hlavní modul tedy nebude čekat na okamžité dokončení práce volaného modulu, ale bude pokračovat dále a pomocí tzv. *Observeru*, bude pravidelně kontrolovat činnost modulu běžícího v pozadí:

```
cliNode = slicer.cli.run(slicer.modules.background_classifier,
                         parameters=cliNodeParameters)
cliNode.AddObserver('ModifiedEvent', self.checkFeaturePreparationStatus)
```

Komunikace mezi moduly lze popsat obrázkem 2.10.



Obr. 2.10: Architektura modulu

2.3.3 Optimalizace

V průběhu práce došlo k výraznému zlepšení času trénování a predikce. Velký vliv na to měla normalizace a standardizace příznakového vektoru, dále také nastavení parametrů klasifikátoru. Rozhodujícím krokem ovšem byla možnost vykonávat operaci klasifikace a následné predikce paralelně. Knihovna *scikit-learn* nabízí implementaci tzv. *BaggingClassifier*, do níž je možno zabalit model SVC. Parametr *n_jobs* pak udává počet paralelních procesů, při hodnotě $n\_jobs = -1$ jsou využita všechna jádra procesoru. Zabalení na úrovní kódu vypadá následovně:

Výpis 2.5: Paralelní klasifikace

```
n_estimators = 16
svc = BaggingClassifier(svm.SVC(max_iter=10000, tol=1,
                                shrinking=0, cache_size=500,
                                verbose=0, kernel='rbf'),
                                max_samples=1.0 / n_estimators,
                                n_estimators=n_estimators,
                                n_jobs=-1)
```

Tabulka 2.7 udává naměřené časy, jež byly měřeny v průběhu řešení práce v rámci segmentace CT obrazových sekvencí. Měření bylo prováděno na trénovací množině o velikost 161280 pixelů, řádek příznakového vektoru obsahoval 50 hodnot. U paralelních operací bylo využito všech jader procesoru (Intel Core i5-7200U, 2 + 2 jádra).

Tab. 2.7: Naměřené časy

Událost	Čas
`Výchozí stav`	35 min
`Normalizace dat`	4,5 min
`Nalezení nejvhodnějších parametrů`	180 s
`Standardizace dat`	97 s
`Paralelní běh`	11,5 s

2.4 Výsledky segmentace

V rámci bakalářské práce byly segmentovány obrazové sekvence jež odpovídají sekvencím z kapitoly 2.2. Jedná se tedy o obrazové sekvence CT, MRI–T1 bez svalu, MRI–T2 bez svalu, MRI–T1 se svalem a MRI–T2 se svalem. Pro nalezení optimálních parametrů bylo použito již několikrát zmíněné techniky *Grid Search* implementované v knihovně *scikit-learn*. V první řade jsou v této kapitole uvedeny grafy popisující průměrné skóre klasifikátoru v závislosti na parametrech *gamma* a *C*. Pro nejúspěšnější pár je vyhodnocen Dice koeficient a je zobrazena výsledná predikovaná maska.

Dice koeficient

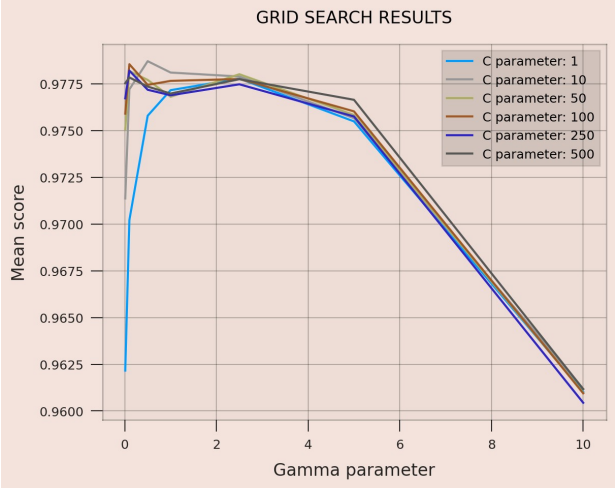
Míra podobnosti dvou obrazů $\mathbf{A}, \mathbf{B}$ lze popsat pomocí Dice koeficientu. Než přejdeme k definici tohoto koeficientu je žádoucí predikované pixely rozdělit do čtyř kategorií.

- N_{TP} – počet *true positive* – skutečně pozitivních pixelů, tedy správně označených jako obratel,
- N_{TN} – počet *true negative* – skutečně negativních pixelů, tedy správně označených jako pozadí,
- N_{FP} – počet *false positive* – falešně pozitivních pixelů, tedy nesprávně označených jako obratel,
- N_{FN} – počet *false positive* – falešně negativních pixelů, tedy nesprávně označených jako pozadí.

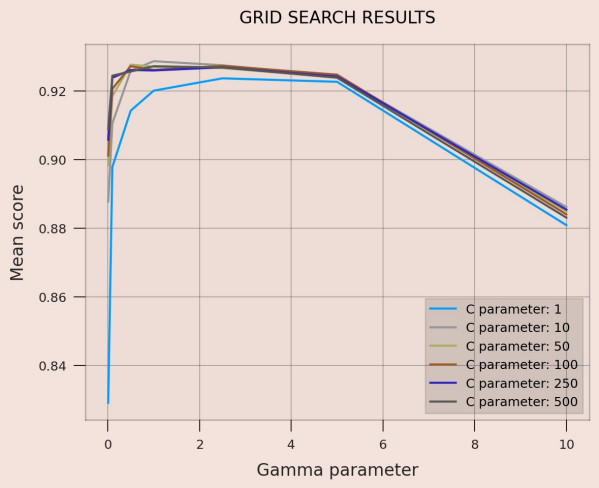
Při výpočtu Dice koeficientu jsou vynechány pixely spadající do kategorie skutečně negativní. Je tak zohledněna situace, kdy v obrazu převládá pozadí. Dice koeficient můžeme popsat následujícím vztahem [37]:

$$D(\mathbf{A}, \mathbf{B}) = \frac{2 \cdot N_{\mathrm{TP}}}{2 \cdot N_{\mathrm{TP}} + N_{\mathrm{FP}} + N_{\mathrm{FN}}} \tag{2.4}$$

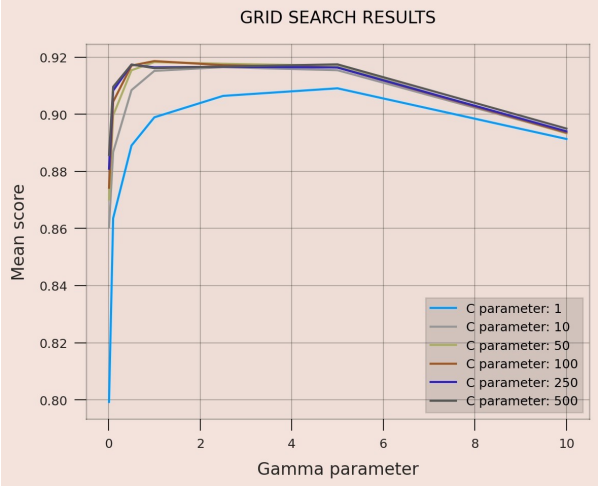
Na následující straně jsou zobrazeny výsledné grafy *Grid Search*.



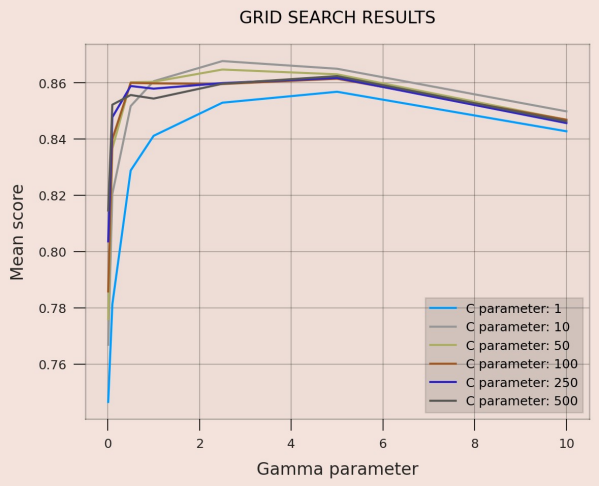
(a) CT



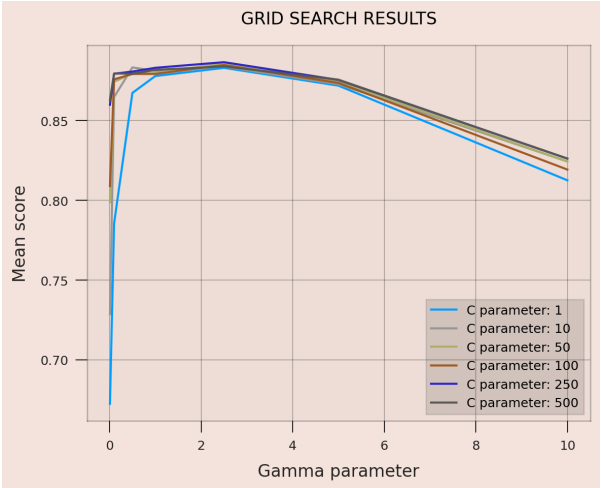
(b) MRI–T1 bez svalu



(c) MRI–T2 bez svalu



(d) MRI–T1 se svalem



(e) MRI–T2 se svalem

Obr. 2.11: Grafy *Grid Search*

V tabulce 2.8 jsou uvedeny hodnoty Dice koeficientu pro každou modalitu, jsou zde také uvedeny nejoptimálnější hodnoty parametrů C a *gamma* a celkový počet pixelů v trénovací a validační množině.

Tab. 2.8: Optimální parametry a Dice koeficient

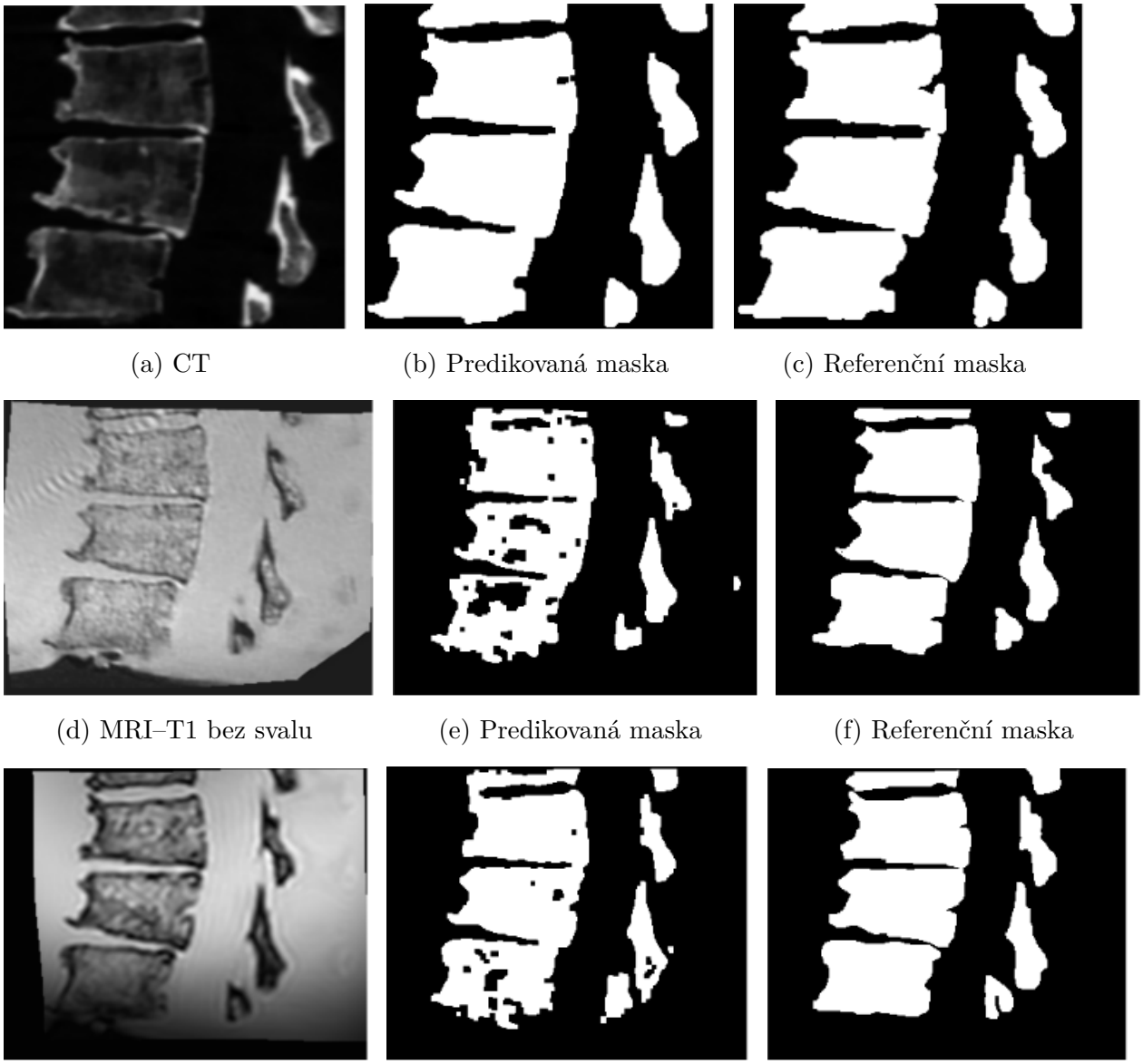
Modalita	Trénovací množina	Validační množina	C	*gamma*	Dice koeficient
`CT`	98782	51920	10	0,5	0,980
`MRI-T1 bez svalu`	161280	80640	10	1	0,948
`MRI-T2 bez svalu`	161280	80640	100	1	0,967
`MRI-T1 se svalem`	54560	27280	10	2,5	0,910
`MRI-T2 se svalem`	116500	58250	250	2,5	0,917

Dále je v tabulce 2.9 uveden čas potřebný pro trénování klasifikátoru a následné predikování pro jednotlivé obrazové sekvence při optimální parametrech C a *gamma*. Časy predikce velmi závisí na vstupních datech. Pro data, jež jsou snadněji separovatelná (tedy zejména kontrastní obrazové sekvence CT), se jedná o časy rychlejší a korespondující s časy trénovací fáze. Pro data, která jsou více zatížena šumem a hůře se separují, čas predikce narůstá poměrně rychle. Na pozadí by navíc mělo probíhat kopírování vnitřní struktury klasifikátoru a převádění dat do vnitřního formátu knihovny *scikit-learn*, respektive *LIBSVM*, jejíž implementace klasifikátoru je v rámci *scikit-learn* využívána. Možností optimalizace by byla redukce počtu podpůrných vektorů, které jsou používány v samotném jádru výpočtu, nicméně tento krok může vést k větší chybovosti. Jelikož se jedná o segmentaci obrazu z medicínského prostředí, byla upřednostněna možnost dosahující menší chybovosti, což na druhé straně vede také k větším časovým nárokům.

Tab. 2.9: Naměřené časové hodnoty

Modalita	Čas - trénovací fáze	Čas - predikce
`CT`	11,5 s	13,1 s
`MRI-T1 bez svalu`	43,7 s	116 s
`MRI-T2 bez svalu`	48,5 s	115,8 s
`MRI-T1 se svalem`	11,5 s	19,6 s
`MRI-T2 se svalem`	28,5 s	68,1 s

Na závěrečných stranách jsou zobrazeny predikované masky pro jednotlivé modality. Pro srovnání je zde uvedena také referenční maska a samotný obraz, pro který byla segmentace prováděna.

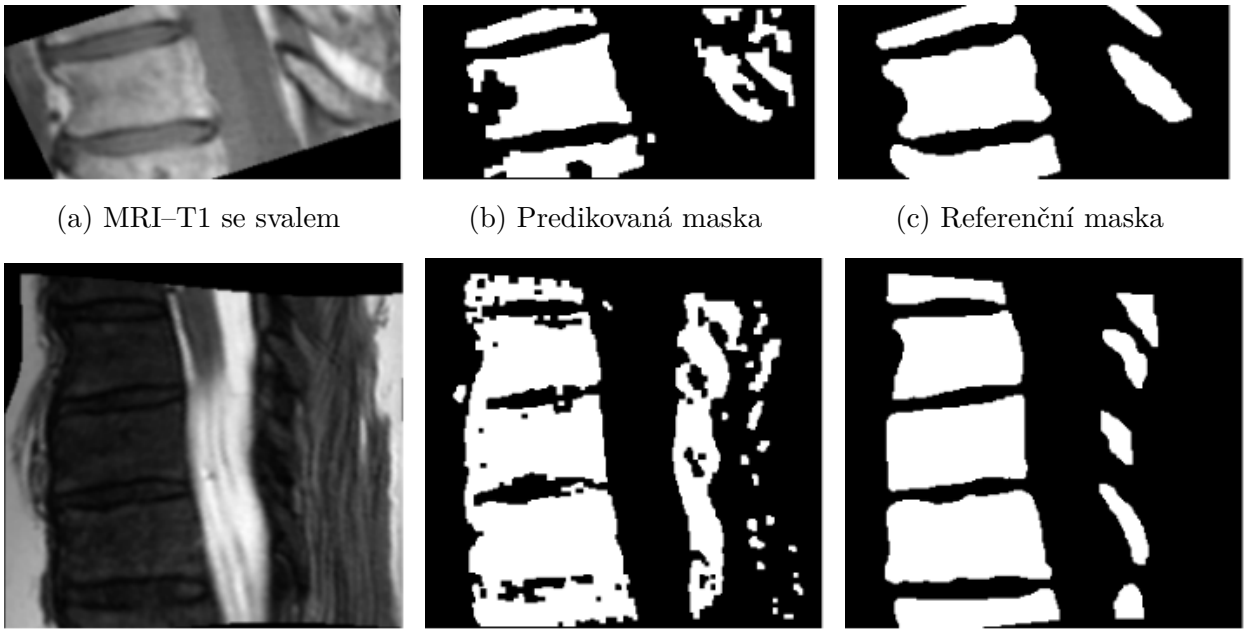


(a) CT (b) Predikovaná maska (c) Referenční maska

(d) MRI–T1 bez svalu (e) Predikovaná maska (f) Referenční maska

(g) MRI–T2 bez svalu (h) Predikovaná maska (i) Referenční maska

Obr. 2.12: Predikované masky

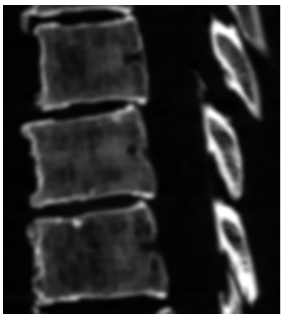


(a) MRI–T1 se svalem (b) Predikovaná maska (c) Referenční maska

(d) MRI–T2 se svalem (e) Predikovaná maska (f) Referenční maska

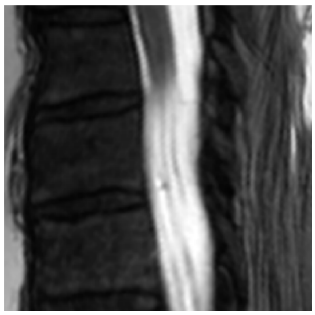
Obr. 2.13: Predikované masky

Poslední informace, která bude v této závěrečné kapitole uvedena, se bude týkat segmentace vybrané obrazové sekvence MRI se svalem, jež je registrována s obrazovou sekvencí CT, a u které budou použity masky právě z příslušné sekvence CT obrazů. Bude tak naplněn původní cíl práce, tedy realizovat segmentaci registrovaného obrazu MRI s nápomocí masek vytvořených pro CT obraz. Bohužel realizace tohoto úkolu, ostatně jak bylo několikrát zmíněno v průběhu práce, je možná jen s obrazovými sekvencemi, pro které registrace přinesla dostatečné výsledky. To ovšem, i přes značné věnování autorova času, nebylo možné u všech sekvencí uskutečnit,jelikož obrazová data MRI nenabývají odpovídající kvality. Na obrázku 2.14 můžeme vidět predikovanou masku pro sekvenci MRI–T2 se svalem, referenční masku a příslušné obrazy MRI a CTI. Tato MRI sekvence vykazovala alespoň zčásti poměrně dobré prostorové srovnání s CT sekvencí. Predikce klasifikátoru pak dosáhla hodnoty Dice koeficientu 0,862. Lepších výsledků je možné dosáhnout segmentací ještě menší části patéře, nejlépe jednotlivého obratle, nicméně poté dochází k velkému časovému nárustu díky pracné přípravě dat. Navíc rozhodujícím faktorem je vždy kvalita vstupních obrazových dat, MRI obrazové sekvenci tento požadavek bohužel splňují jen částečně.

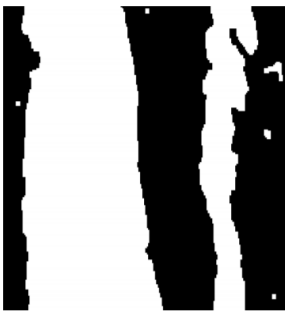


(a) CT obraz

(b) Referenční maska CT použitá pro MRI obraz



(c) Registrovaný MRI–T2 obraz



(d) Predikovaná maska

Obr. 2.14: Predikovaná maska pro MRI obraz

Závěr

Jádrem této bakalářské práce bylo vytvoření rozšiřujícho modulu pro program 3D Slicer implementovaného v jazyce Python. Modul v sobě sdružuje přípravu vstupních dat, extrakci příznaků a provedení samotné segmentace vstupních obrazových dat pomocí modelu strojového učení SVM. Uživatelské rozhraní modulu je organizováno do tří částí, jež odpovídají fázi zpracování dat. Uživateli je umožněno perzistentně ukládat výsledky své práce v podobě serializovaných objektů. Uživatel tedy může průběžně ukládat příznakový vektor, popřípadě tento vektor rozšiřovat dalšími extrahovanými příznaky, dále je pak uživatel nabízeno uložit natrénovaný klasifikátor. Samotná logika programu pak využívá možnost plné paralelizace trénování a predikce klasifikátoru, časová optimalizace je popsána v tabulce 2.7. Možné zlepšení autor vidí v lepší integraci do programu Slicer, používání interních funkcí a samotnou spjatost s kódem (tzv. *codebase*) programu Slicer. I přestože byl kód modulu a samotných skriptů několikrát revidován, je zde pořád velký prostor pro zlepšení, zejména pak v psaní srozumitelného čistého kódu.

Jako vstupní data byly použity obrazové sekvence páteře pořízené pomocí CT a MRI. Data byla poskytnuta lékaři z Fakultní nemocnice Brno Bohunice. V průběhu práce byla s lékaři alespoň částečně udržována komunikace, lékařům byly zaslány ruční segmentace obrazů s žádostí o vyjádření, popřípadě částečnou opravu, dále pak celkový obraz páteře, tak aby byly jednotlivé obratle anotamicky popsány a očíslovány. Bohužel vlivem kybernetických útoků, které byly cíleny právě na nemocniční počítače, a vlivem pozdější pandemie byla komunikace ukončena. Obrazová MRI data byla registrovaná s vybranou obrazovou sekvencí CT tak, aby bylo možné později použít jednotně segmentační masky a využít tak kontrastu obrazů CT. Tento úkol byl splněn jen částečně, důvodem bylo již několikrát zmíněna kvalita dat, vůbec počáteční prostoré srovnání obrazů vyžadovalo značné usílí. Kompromisem řešení bylo převzorkování obrazů MRI, částečná registrace obrazů a možnost provedení alespoň hrubého vizuálního porovnání obrazů. K vyhodnocení úspěšnosti segmentace byl použit Dice koeficent, nejlepšího výsledku bylo dosáhnuto u kontrastních obrazů CT, kde díky naleznutí optimálních parametrů klasifikátoru *gamma* a *C* pomocí metody *Grid Search*, implementaci extrakce vhodných příznaků, ekvalizaci histogramu a v neposlední řadě vyhlazení predikované masky pomocí morfologických operací, bylo dosáhnuto Dice koeficientu o hodnotě 0,980.

Přinos práce autor vidí hlavně v samotném vytvoření modulu realizující segmentaci pomocí SVM v rámci programu 3D Slicer. Tento modul může být používán na segmentaci jiných obrazových dat, není nikterak vázán na data používána v této práci, modul má tedy charakter nástroje. Do budoucna by modul mohl být rozšířen o extrakci dalších příznaků, být lépe optimalizován a rozšířen o další funkcionalitu.

Literatura

[1] FAKHRE, Alam, UR RAHMAN Sami, DIN Aziz a Qayum FAWAD. *Medical image registration: Classification, applications and issues* [online]. Journal of Postgraduate Medical Institute, 2008. 32s. [cit. 2019-11-16] Dostupné z URL: <https:www.researchgate.netpublication320486228_Medical_image_registration_in_image_guided_surgery_Issues_challenges_and_research_opportunities>

[2] KALENDER, Willi Alfred. *Computed Tomography: Fundamentals, System Technology, Image Quality, Applications.* Munich: Publicis MCD Verlag, 2000. 220s. ISBN 3-89578-081-2

[3] ŠONKA, Milan a Václav HLAVÁČ. *Počítačové vidění.* Praha: Grada, 1992. 272s. ISBN 80-85424-67-3.

[4] JAN, Jiří. *Medical image processing, reconstruction, and restoration: concepts and methods.* Boca Raton, FL, USA: CRC Press, Taylor and Francis Group, 2006. 760s. ISBN: 0-8247-5849-8.

[5] MAINTZ, Antoine a Max VIERGEVER. *A survey of medical image registration.* [online] Medical Image Analysis 2, č.1, 1998. s.1-36. [cit. 2019-11-17] ISSN: 0924-3275. Dostupné z URL: <http://scholar.google.cz/scholar_url?url=https://dspace.library.uu.nl/bitstream/handle/1874/18921/maintz_98_an_overview.pdf%3Fsequence%3D2&hl=cs&sa=X&scisig=AAGBfm3TrwKwkK-1735eecuvdfZNdme4MQ&nossl=1&oi=scholarr>

[6] MANI, V.R.S a Selvaraj ARIVAZHAGAN. *Survey of Medical Image Registration* [online]. Journal of Biomedical Engineering and Technology, 2013. s.8-25. [cit. 2019-11-17] Dostupné z URL: <http://pubs.sciepub.com/jbet/1/2/1/index.html>

[7] BOTTEMA, Oene a Bernard ROTH. *Theoretical Kinematics.* New York: North-Holland Pub, 1979. ISBN 0-486-66346-9.

[8] ANGENENT, Sigurd, PICHON, Eric a Allen TANNENBAUM. *Mathematical methods in medical image processing* [online]. Bulletin (new series) of the American Mathematical Society, 2006. 43. s.365-396. [cit. 2019-11-19] doi: 10.1090/S0273-0979-06-01104-9. Dostupné z URL: <https://www.researchgate.net/publication/236639847_Mathematical_methods_in_medical_image_processing>

[9] LACMANOVÁ, Zdeňka. *Obrazová registrace medicínských dat* [online]. Praha, 2013. [cit. 2019-11-20]. Bakalářská práce. Univerzita Karlova, Matematicko-fyzikální fakulta. Vedoucí práce Jindřich Soukup. Dostupné z URL: <https://is.cuni.cz/webapps/zzp/detail/130394/>

[10] RUECKERT, Daniel, LEACH, Martin, HAWKES, David et al. *Nonrigid registration using free-form deformations: application to breast MR images* [online]. IEEE Transactions on Medical Imaging, 18, č.8, 1999. s. 712-721. [cit. 2019-11-20] doi: 10.1109/42.796284. Dostupné z URL: <https://ieeexplore.ieee.org/document/796284/authors>

[11] KYBIC, Jan a Michael UNSER. *Fast parametric elastic image registration* [online]. IEEE Transactions on Image Processing, 12, č. 11, 2003. s.1427-1442. [cit. 2019-11-22] Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1240109&isnumber=27803>

[12] HLAVÁČ, Václav. *Jasové a geometrické transformace* [online]. ČVUT, Český institut informatiky, robotiky a kybernetiky, Praha, 2003. [cit. 2019-11-22] Dostupné u URL: <http://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/18BrightGeomTxCz.pdf>

[13] ZITOVÁ, Barbara a Jan Flusser. *Image registration methods: a survey* [online]. [cit. 2019-11-23] Image and vision computing, 21, č.11, 2003. s.977-1000. Dostupné z URL: <https://www.researchgate.net/profile/Jan_Flusser/publication/222648347_Image_Registration_Methods_A_Survey/links/5ba9f31c92851ca9ed238b48/Image-Registration-Methods-A-Survey.pdf>

[14] LEHMANN, Thomas M., GÖNNER, Claudia a Klaus SPITZER. *Survey: Interpolation Methods in Medical Image Processing* [online]. IEEE Transactions on medical imaging, 18, č.11, 1999. s.1049-1076. [cit. 2019-11-25] Dostupné z URL: <http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/Digital_Image_Processing_2ndEd.pdf>

[15] GONZALEZ, Rafael C. a Richard E. WOODS. *Digital image processing [2nd ed.]*. Upper Saddle River: Prentice Hall, 2002. 793 s. ISBN 0-201-18075-8.

[16] ŠPANĚL, Michal a Vítězslav BERAN. *Obrazové segmentační techniky. Přehled existujících metod* [online]. Brno, 2006. [cit. 2019-11-25] Vysoké učení technické v Brně. Fakulta informačních technologií. Dostupné z URL: <http://www.fit.vutbr.cz/~spanel/segmentace/>

[17] KOBAN, Martin. *Segmentace míšního kanálu a meziobratlových plotének v MRI datech* [online]. Brno, 2018 [cit. 2019-11-26]. Bakalářská práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav biomedicínského inženýrství. Vedoucí práce Roman Jakubíček. Dostupné z URL: <http://hdl.handle.net/11012/82817>

[18] AL-SHARIF, Sharif, DERICHE Mohammed a Nabil MAALEJ. *Medical images Segmentation Using WGAC Technique.* Saarbrücken: VDM Publishing, 2010. 88s. ISBN: 978-3-639-26547-7

[19] SHAPIRO, Linda a George STOCKMAN. *Computer vision.* [online] Upper Saddle River, NJ: Prentice Hall, 2001, [cit. 2019-12-26]. ISBN 0130307963. Dostupné z URL: <http://nana.lecturer.pens.ac.id/index_files/referensi/computer_vision/Computer%20Vision.pdf

[20] OTSU, Nobuyuki. *A threshold selection method from gray-level histograms* [online]. IEEE Transactions on Systems, Man and Cybernetics, 9 , č.1, 1979. s.62–66. Brno, 2006. [cit. 2019-11-26] DOI:10.1109/TSMC.1979.4310076. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4310076&isnumber=4310064

[21] BEUCHER, Serge a Christian LANTUÉJOUL. *Use of watersheds in contour detection.* [online]. International Workshop on image processing, Rennes, France, 1979. 12s. Brno, 2006. [cit. 2019-11-27] Dostupné z URL: <http://www.cmm.mines-paristech.fr/~beucher/publi/watershed.pdf>

[22] HARIS, Kostas, Nicos MAGLAVERAS a Agelos KATSAGGELOS. *Hybrid image segmentation using watersheds and fast region merging. IEEE Transactions on Image Processing* [online]. 7(12), 1684-1699 [cit. 2019-11-27]. DOI: 10.1109/83.730380. ISSN 10577149. Dostupné z URL: <http://ieeexplore.ieee.org/document/730380/>

[23] KASS, Michael, Andrew WITKIN a Demetri TERZOPOULOS. *Snakes: Active contour models. International Journal of Computer Vision* [online]. 1988, 1(4), 321-331 [cit. 2019-11-30]. DOI: 10.1007/BF00133570. ISSN 0920-5691. Dostupné z URL: <http://www.cs.ait.ac.th/~mdailey/cvreadings/Kass-Snakes.pdf>

[24] CHOLLET, François. *Deep learning v jazyku Python: knihovny Keras, Tensorflow.* Praha: Grada, 2019. Knihovna programátora (Grada). ISBN 978-80-247-3100-1.

[25] NEUBAUER, Jiří, Marek SEDLAČÍK a Oldřich KŘÍŽ. *Základy statistiky: Aplikace v technických a ekonomických oborech.* 2.vydání. Praha: Grada, 2012. 280s. ISBN 978-80-247-5786-5.

[26] CORTES, Corinna a Vladimir VAPNIK. *Support-vector networks. Machine Learning* [online]. 1995, 20(3), 273-297 [cit. 2019-11-30]. DOI: 10.1007/BF00994018. ISSN 0885-6125. Dostupné z URL: <http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf>

[27] HEARST, Marti, Susan DUMAIS, Edgar OSUNA et al. *Support vector machines. IEEE Intelligent Systems and their Applications* [online]. 1998, 13(4), 18-28 [cit. 2019-11-30]. DOI: 10.1109/5254.708428. ISSN 1094-7167. Dostupné z URL: <http://ieeexplore.ieee.org/document/708428/>

[28] BURGES, Christopher. *A Tutorial on Support Vector Machines for Pattern Recognition*[online]. In Data Mining and Knowledge Discovery, 1998. s. 121 – 167. [cit. 2019-11-30]. Dostupné na URL: <http://research.microsoft.com/pubs/67119/svmtutorial.pdf>

[29] EVGENIOU, Theodoros a Pontil MASSIMILIANO. *Support Vector Machines: Theory and Applications*[online]. Springer Berlin Heidelberg, 2001, 2001-9-20, s. 249-257 [cit. 2019-11-30]. ISBN 978-3-540-42490-1. <https://www.researchgate.net/publication/221621494_Support_Vector_Machines_Theory_and_Applications>

[30] SHELHAMER, Evan, Jonathan LONG a Trevor DARRELL. *Fully convolutional networks for semantic segmentation*[online]. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2015, s.3431-3440. [cit. 2019-12-01].ISBN 978-1-4673-6964-0. Dostupné z URL: <http://ieeexplore.ieee.org/document/7298965/>

[31] YAMASHITA, Rikiya, Mizuho NISHIO, Richard KINH GIAN DO a Kaori TOGASHI. *Convolutional neural networks: an overview and application in radiology*[online]. Insights into Imaging. Springer, 2018, (9), s.611-629. [cit. 2019-12-01]. DOI:10.1007/s13244-018-0639-9. Dostupné z URL: <https://www.semanticscholar.org/paper/Convolutional-neural-networks%3A-an-overview-and-in-Yamashita-Nishio/7580c72df96488927c59a5f31a4b10784adfc749>

[32] RONNEBERGER, Olaf, Philipp FISCHER a Thomas BROX. *U-net: Convolutional networks for biomedical image segmentation* [online]. International Conference on Medical Image Computing and Computer-Assisted Intervention. Cham: Springer International Publishing, 2015. [cit. 2019-12-01]. ISBN 978-3-319-24573-7. Dostupné z URL: <https://arxiv.org/abs/1505.045970>

[33] GLOROT, Xavier, Antoine BORDES a Yoshua BENGIO. *Deep sparse rectifier neural networks.*[online]. Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, s. 315-323. [cit. 2019 12-02]. Dostupné z URL: <http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf>

[34] FEDOROV, Andriy, Reinhard BEICHEL,Julien FINET et al. *3D Slicer as an image computing platform for the Quantitative Imaging Network* [online]. Magnetic resonance imaging, 30, č.9, 2012. s.1323-1341. [cit. 2019 12-03]. Dostupné z URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3466397/>

[35] KORČUŠKA, Robert. *Segmentace tomografických dat v prostředí 3D Slicer* [online]. Brno, 2015. [cit. 2019 12-03]. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav telekomunikací. Vedoucí Jan Mikulka. Dostupné z URL: <http://hdl.handle.net/11012/39944>

[36] PILGRIM, Mark. *Ponořme se do Python(u) 3.* Praha: CZ.NIC, z.s.p.o., 2010. ISBN 978-80-904248-2-1.

[37] TAHA, Abdel a Allan HANBURY. *Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool*[online]. In BMC Medical Imaging 2015, [cit.2019-12-8]. DOI: 10.1186/s12880-015-0068-x. ISSN 1471-2342. Dostupné z URL: <http://bmcmedimaging.biomedcentral.com/articles/10.1186/s12880-015-0068-x>

Seznam symbolů, veličin a zkratek

MRI	zobrazovaní magnetickou rezonancí – Magnetic Resonance Imaging
CT	výpočetní tomografie – Computed Tomography
SVM	metoda podpůrných vektorů – Support Vector Machines
DICOM	Digital Imaging and Communications in Medicine
HU	Hounsfieldovy jednotky – Hounsfield Units
CNN	konvoluční neuronové sítě – Convolutional Neural Network
ReLU	typ lineární aktivační funkce – Rectified Linear Unit
GUI	grafické uživatelské prostředí – Graphical User Interface
MVC	typ softwarové architektury – Model–View–Controller
MRML	značkovací jazyk – Medical Reality Markup Language
API	aplikační rozhraní – Application Programming Interface
XML	značkovací jazyk – Extensible Markup Language
ROI	oblast zájmu – Region Of Interest
CLI	rozhraní příkazové řádky – Command Line Interface
UI	uživatelské rozhraní – User Interface
nrrd	formát obrazových dat – Nearly Raw Raster Data
png	formát obrazových dat – Portable Network Graphics

Obsah adresáře

/
- `Vybrané obrazové sekvence`
- `Obrazové sekvence pro testování modulu`
- `Návod ke spuštění skriptů`
- `Návod k instalaci modulu`
- `Skripty` ..... Složka s nativními Python skripty
- `SVM Slicer` ..... Složka s implementací modulu
- `3D Slicer` ..... Složka obsahující instalační soubor programu 3D Slicer